PB89-220008

ANNIE-IDE, a System for Developing Interactive User Interfaces for Environmental Models (Programmers Guide)

AQUA TERRA Consultants, Decatur, GA

Prepared for:

Environmental Research Lab., Athens, GA

Apr 89

U.S. Department of Commerce
National Technical Information Service
NTIS

United States
Environmental Protection
Agency

Environmental Research
Laboratory
Athens GA 30613

EPA/600/3-89/034
April 1989

PB89-220008

Research and Development

EPA

ANNIE-IDE, A System for Developing Interactive User Interfaces for Environmental Models

(Programmers Guide)

REPRODUCED BY
U.S. DEPARTMENT OF COMMERCE
NATIONAL TECHNICAL INFORMATION SERVICE
SPRINGFIELD, VA. 22161

EPA/600/3-89/034
April 1989

ANNIE-IDE, A SYSTEM FOR DEVELOPING INTERACTIVE USER INTERFACES FOR ENVIRONMENTAL MODELS

(PROGRAMMERS GUIDE)

by

John L. Kittle, Jr.
Paul R. Hummel
John C. Imhoff

AQUA TERRA Consultants
Decatur, Georgia 30030

Contract Number 68-03-3513

Project Officer

Thomas O. Barnwell, Jr.
Assessment Branch
Environmental Research Laboratory
Athens, Georgia

ENVIRONMENTAL RESEARCH LABORATORY
OFFICE OF RESEARCH AND DEVELOPMENT
U.S. ENVIRONMENTAL PROTECTION AGENCY
ATHENS, GEORGIA 30613

DISCLAIMER

The information in this document has been funded wholly or in part by the United States Environmental Protection Agency under Contract No. 68-03-3513 with AQUA TERRA Consultants. It has been subject to the Agency's peer and administrative review, and it has been approved for publication as an EPA document. Mention of trade names or commercial products does not constitute endorsement or recommendation for use by the U.S. Environmental Protection Agency.

TECHNICAL REPORT DATA

(Please read Instructions on the reverse before completing)

1. REPORT NO.
EPA/600/3-89/034

2.

3. RECIPIENT'S ACCESSION NO.
PB89 220008/AS

4. TITLE AND SUBTITLE
ANNIE-IDE, A SYSTEM FOR DEVELOPING INTERACTIVE USER INTERFACES FOR ENVIRONMENTAL MODELS (PROGRAMMERS GUIDE)

5. REPORT DATE
April 1989

6. PERFORMING ORGANIZATION CODE

7. AUTHOR(S)
John L. Kittle, Jr., Paul R. Hummel, and John C. Imhoff

8. PERFORMING ORGANIZATION REPORT NO.

9. PERFORMING ORGANIZATION NAME AND ADDRESS
AQUA TERRA Consultants
Decatur GA 30030

10. PROGRAM ELEMENT NO.
CCBE1A

11. CONTRACT/GRANT NO.

12. SPONSORING AGENCY NAME AND ADDRESS
Environmental Research Laboratory - Athens, GA
Office of Research and Development
U.S. Environmental Protection Agency
Athens GA 30613

13. TYPE OF REPORT AND PERIOD COVERED

14. SPONSORING AGENCY CODE
EPA/600/01

15. SUPPLEMENTARY NOTES

16. ABSTRACT

The This document is a guide to the newly developed computer software called ANNIE-IDE, the ANNIE Interaction Development Environment. ANNIE-IDE provides a consistent methodology for building interactive interfaces for environmental software. ANNIE-IDE combines a toolkit of utility subroutines for building individual interactive screens with instructions for developing two parallel products: a file containing all text , questions, and messages used in interactive communication and a Fortran program containing the control strategy and sequencing instructions for interactions. The system provides an efficient means for storing and updating screen contents.

The ANNIE-IDE package is based on a re-evaluation of currently available tools and ideas for constructing a user interface. Accordingly, the manual draws on an expanding body of knowledge of the "human factors" involved in effective design of nan-machine interfaces. By providing a straightforward but powerful set of tools, the ANNIE-IDE software and manual greatly reduce the difficulty a programmer will encounter in developing an interactive user interface.

17. KEY WORDS AND DOCUMENT ANALYSIS

a. DESCRIPTORS	b. IDENTIFIERS/OPEN ENDED TERMS	c. COSATI Field/Group

18. DISTRIBUTION STATEMENT
RELEASE TO PUBLIC

19. SECURITY CLASS *(This Report)*
UNCLASSIFIED

20. SECURITY CLASS *(This page)*
UNCLASSIFIED

21. NO. OF PAGES
180

22. PRICE
PC $21.95 / MF $6.95

EPA Form 2220-1 (Rev. 4-77) PREVIOUS EDITION IS OBSOLETE

FOREWORD

As environmental controls become more costly to implement and the penalties of judgement errors become more severe, environmental quality management requires more efficient analytical tools based on greater knowledge of the environmental phenomena to be managed. As part of this Laboratory's research on the occurrence, movement, transformation, impact, and control of environmental contaminants, the Assessment Branch develops management or engineering tools to help pollution control officials address environmental problems.

Recent progress in environmental science and engineering has seen the increasing use of interactive interfaces for computer models. Initial applications centered on the use of interactive functions to assist the modeler in building complicated input sequences required by batch programs. From these original applications, interactive capabilities are rapidly being expanded to provide "smarter" and more flexible communication between users and the environmental models and databases they use. This manual provides a straightforward, consistent methodology for programmers who have the task of designing and implementing new interactive interface systems.

Rosemarie C. Russo, Ph.D.
Director
Environmental Research Laboratory
Athens, Georgia

ABSTRACT

This document is a guide to the newly developed computer software called ANNIE-IDE - the ANNIE Interaction Development Environment. ANNIE-IDE provides a consistent methodology for building interactive interfaces for environmental software. ANNIE-IDE combines a toolkit of utility subroutines for building individual interactive screens with instructions for developing two parallel products: a file containing all text, questions, and messages used in interactive communication, and a Fortran program containing the control strategy and sequencing instructions for interactions. The system provides an efficient means for storing and updating screen contents.

The ANNIE-IDE package is based on a re-evaluation of currently available tools and ideas for constructing a user interface. Accordingly, the manual draws on an expanding body of knowledge of the "human factors" involved in effective design of man-machine interfaces. By providing a straightforward, but powerful set of "tools" and a detailed consideration of the best way to use these tools, the ANNIE-IDE software and manual greatly reduce the difficulty a programmer will encounter in developing an interactive user interface.

This report was submitted in fulfillment of Work Assignment No. 2(B) of Contract No. 68-03-3513 by AQUA TERRA Consultants under the sponsorship of the U.S. Environmental Protection Agency. This report covers a period from November 1987 to September 1988, and work was completed as of September 1988.

CONTENTS

Disclaimer ii
Foreword iii
Abstract iv
Figures viii
Tables x
Acknowledgments xi

1. Introduction 1
  1.1 Who Should Use ANNIE-IDE? 2
  1.2 Why Use ANNIE-IDE? 3
  1.3 Manual Contents 3

2. Overview 4
  2.1 Background 4
    2.1.1 WDM and ANNIE 4
    2.1.2 EMIFE 4
    2.1.3 ANNIE-IDE 6
  2.2 Overview of Toolkit and Methodology 6
    2.2.1 Screen Content Construction 7
    2.2.2 Screen Interaction Control 8
    2.2.3 Housekeeping 8
    2.2.4 Sample Screens 8
  2.3 System Requirements and Goals 12
  2.4 Limitations 13
  2.5 Toolkit Demonstration Examples 14

3. Theoretical Considerations of ANNIE-IDE Development . . . 16
  3.1 General Theory of Interactive Interfaces 16
    3.1.1 Human Factors Goals 17
    3.1.2 Why Interactive Systems Fail 17
    3.1.3 Rules of System Design 18
    3.1.4 Styles for Interactive Systems 18
    3.1.5 Menu Selection Interactions 19
    3.1.6 Rules for Grouping Items in Menus and Tree Structures 19
    3.1.7 Menu Item Presentation Sequence 19
    3.1.8 General Considerations 20
    3.1.9 Form Fill-in Interactions 20
  3.2 Application of Human Factors Theory to ANNIE-IDE . . 21
  3.3 Screen Layout 22
  3.4 Data Window 24
  3.5 Assistance Window 24

3.6 Instruction Window 28
3.7 Command Line 28
3.8 User Control Within Screens 30
  3.8.1 User Interaction Modes 30
  3.8.2 Keystrokes and Communication Features . . . 31
  3.8.3 Movement Between Interaction Modes 33
3.9 Screen Definition Directives 33
3.10 Screen Sequencing 35

4. Instructions for Using ANNIE-IDE 36
4.1 Building and Storing Screen Definition Files . . . 36
  4.1.1 Text Screen Directives 40
  4.1.2 Menu Screen Directives 41
  4.1.3 1-D Array (PRM1) Screen Directives 42
  4.1.4 2-D Array (PRM2) Screen Directives 43
  4.1.5 File Screen Directives 44
  4.1.6 Using MESSIE 45
  4.1.7 Directives to Provide Assistance to User . . 47
4.2 Designing and Developing the Interactive Program . 48
  4.2.1 Initialize ANNIE-IDE 48
  4.2.2 Initialize Parameters 48
  4.2.3 Read Existing Parameter Values From a File . 49
  4.2.4 Update Parameter Values 50
  4.2.5 Write Revised Parameter Values To File . . . 50
  4.2.6 Shut Down ANNIE-IDE 50
  4.2.7 Using Toolkit Subroutines for Screen Interaction Control 50
  4.2.8 Closing Comments on the Development Process 57
4.3 Advanced Topics 58
  4.3.1 User System Specifications File (TERM.DAT) . 58
  4.3.2 Interscreen Movement Commands 59
  4.3.3 Control of Assistance Window Containing System Status 59

5. Example Applications 60
5.1 Initialize System 60
5.2 Initialize Model Parameters 61
5.3 Read File Input For a Model 61
5.4 Display Text 62
5.5 Display Text With Numeric Information 64
5.6 Select an Item From a Menu 65
5.7 Open a File 67
5.8 Edit One-dimensional Array of Data 68
5.9 Edit Two-dimensional Array of Data 70
5.10 Enter Two-dimensional Array of Data with Buffer Use 72
5.11 Set Limits and Defaults for One or Two-dimensional Data 74

5.12 Display System Status Information 75
5.13 Set Command Availability 76
5.14 Save Current Menu Text 77
5.15 Determine User Screen Exit Command 78
5.16 Write File Containing Model Parameters 78
5.17 Shut Down System 79

6. Recommendations for ANNIE-IDE Enhancements 80

References 82

Appendices

A. Glossary of Terms 83

B. Reference Manual 85
B.1 Frequently Called Routines 86
B.2 Summary of All Toolkit Routines 138
B.3 Data Structures 144
B.4 Screen Definition Directives 155
B.5 File Formats 158
B.6 TERM.DAT Parameters 159
B.7 ANNIE-IDE Commands 161
B.8 Color Codes for PC Color Implementation 162
B.9 Distribution Disk Contents 163

FIGURES

2.1 ANNIE functions and capabilities. 5
2.2 ANNIE-IDE toolkit. 7
2.3 Sample text screen. 9
2.4 Sample menu screen. 10
2.5 Sample 1-D array screen. 10
2.6 Sample 2-D array menu screen. 11
2.7 Sample file screen. 11
3.1 Screen format utilized by ANNIE-IDE. 23
3.2 Example error message in instruct window. 28
3.3 Screen layout in assist mode. 30
3.4 Screen directive functions. 34
4.1 Building an interactive application 37
4.2 Building and storing a screen definition file. 38
4.3 Template screen definition file. 39
4.4 Structure of interactive program providing user interface to an environmental model. 49
4.5 Structure chart for text display subroutines. 52
4.6 Text display subroutines. 53
4.7 Structure chart for menu subroutines. 53
4.8 Structure chart for 1-D array subroutines. 54
4.9 One-dimensional array input subroutines. 55
4.10 Structure chart for 2-D array edit subroutines. 56
4.11 Two-dimensional data edit subroutines. 56
4.12 Structure chart for file subroutines. 57
4.13 File management subroutines. 57
4.14 Sample TERM.DAT file. 58
5.1 Fortran code to initialize ANNIE-IDE. 60
5.2 Fortran code to initialize model parameters. 61
5.3 Fortran code to read model parameters from a file. . . 62
5.4 Fortran code to display text. 62
5.5 Screen definition directives to display text. 63
5.6 Resulting text screen. 63
5.7 Fortran code to display text with numeric information. 64
5.8 Screen definition directives to display text with numeric information. 64
5.9 Resulting text screen with numeric information. 65
5.10 Fortran code to select from menu. 65
5.11 Screen definition directives to select from menu. . . . 66
5.12 Resulting menu screen. 66
5.13 Fortran code to specify a file. 67
5.14 Screen definition directives to specify a file. 67
5.15 Resulting file specification screen. 68

5.16 Fortran code to enter 1-D data. 68
5.17 Screen definition directives to enter 1-D data. 69
5.18 Resulting 1-D data screen. 70
5.19 Fortran code to enter 2-D data. 70
5.20 Screen definition directives to enter 2-D data. 71
5.21 Resulting 2-D data screen. 72
5.22 Fortran code to enter 2-D data using buffer. 72
5.23 Screen definition directives to enter 2-D data with use of buffer. 73
5.24 Resulting 2-D data screen with use of buffer. 74
5.25 Fortran code to set limits for data fields. 74
5.26 Fortran code to add status information. 75
5.27 Resulting status display. 75
5.28 Fortran code to set command availability. 76
5.29 Resulting command line display. 76
5.30 Fortran code to save menu text. 77
5.31 Resulting screen with additional text information. . . 77
5.32 Fortran code to determine user screen exit command. . . 78
5.33 Fortran code to write model parameters to a file. . . . 79
5.34 Fortran code to shut down ANNIE-IDE. 79

TABLES

3.1 Advantages and Disadvantages of Candidate Interaction Styles 18
3.2 Assistance Window Types 26
3.3 Commands for Basic Application of ANNIE-IDE 29
3.4 Keystrokes and Their Results 32
3.5 Optional Commands for Interscreen Movement 35
4.1 TEXT Screen Definition Directives 40
4.2 MENU Screen Definition Directives 41
4.3 PRM1 Screen Definition Directives 42
4.4 PRM2 Screen Definition Directives 43
4.5 FILE Screen Definition Directives 44
4.6 Screen Directives for Specifying Assistance 47

ACKNOWLEDGMENTS

Development of the ANNIE-IDE package was made possible by the efforts and support of a number of individuals and organizations. Thomas Barnwell was the EPA Project Monitor during the course of this work. His guidance and support contributed to the successful completion of this project and is gratefully acknowledged. Others from the Environmental Research Laboratory, Athens, GA, who contributed ideas include Robert Carsel and Dr. Lawrence Burns of the EPA and Patrick Cannon, Ronnie Moon and David Disney of Computer Sciences Corporation.

The United States Geological Survey is acknowledged for providing assistance in developing many of the software components utilized by ANNIE-IDE. Dr. Alan Lumb and Kathleen Flynn have had key involvement in developing ANNIE/WDM, from which many of the "tools" in the ANNIE-IDE toolkit have been extracted. During the course of this project, their comments and those of their co-worker Lewis DeLong have allowed us to produce a better product.

Guidance for developing effective strategies for human-computer interaction in ANNIE-IDE was provided by Dr. Albert Badre of the Georgia Institute of Technology who served as a technical consultant for the duration of this project. Dr. Ben Shneiderman's book Designing the User Interface was also a key reference for resolving "human factors" issues. Their ideas and comments have been invaluable.

James Pilotte of General Software Corporation provided the EPA and AQUA TERRA with the computer code and documentation for the Environmental Model Input File Editor (EMIFE). Some of the user interaction screens developed for ANNIE-IDE were based on prototypes contained in EMIFE.

For AQUA TERRA Consultants, Jack Kittle was the Project Manager and was responsible for designing ANNIE-IDE. Paul Hummel had primary responsibility for coding and testing the toolkit subroutines. John Imhoff developed the system documentation. Anthony Donigian and Brian Bicknell provided reviews and comments.

SECTION 1

INTRODUCTION

Over the past three to four years, the implementation and use of interactive interfaces for computer models of all types have become increasingly popular. In the fields of environmental science and engineering the first attraction to interactive functions centered on their ability to assist the modeler in the effort of building complicated, format-dependent input sequences for batch programs. Hence early interactive interfaces often focused on prompting the model user for all the input instructions and model parameter values needed to produce a complete machine-generated batch input sequence. This was a limited application of interactive capability, but nonetheless was an effective means of reducing user error related to formatting and/or omission of necessary input information. The primary benefit of this application was reduced run failure. A natural next step to the extension of interactive capabilities was to provide checking for input model parameter values to assure they fell within a reasonable range. While the concept of checking the reasonableness of input values is valid, the effective implementation of this capability for environmental models has been hindered by the lack of sound scientific information on some model parameters and the very broad range of possible values for others. Value checking, however, can provide a safeguard against the entry of mistaken values for many parameters, and consequently has become an expected function of many interactive systems.

From these original functions, interactive capabilities are rapidly expanding to provide "smarter" and more flexible communication between users and environmental models and/or data bases. Desirable interactive capabilities, which are currently in various stages of implementation, include: (1) "help screens" that offer the user guidance in executing the interactive system or supplying input data, (2) enhanced system logic that allows the user better control of command and operation sequencing, (3) enhanced system checks that assure that the instructions provided by the user are logically consistent, and (4) system capability to adjust the nature of user interaction to accommodate the level of expertise of individual users.

As is often the case with rapidly developing technologies, the approaches and methods that have been used to produce the array of currently available interactive systems vary widely and reflect the knowledge, philosophies, and personal biases of their developers. The diversity in systems capabilities and approaches has resulted in user confusion and frustration and has somewhat limited the effectiveness of the available systems. At the same time, the lack of any sort of guidance or standardized methods for interactive system development has placed an unnecessary burden on programmers given the task of designing and implementing new systems.

The impetus for developing this manual, and the programmer's toolkit (ANNIE-IDE) it describes, is recognition that a need exists for an straightforward, consistent methodology for building interactive interfaces for available and developing environmental models and data bases. An immediate goal is to begin a standardization process for implementing interactive interfaces for environmental models commonly used by the EPA and its contractors. The ANNIE-IDE system is a product of re-evaluation of currently available tools and ideas for constructing a user interface. The system combines a "toolkit" of subroutines for interaction construction that can be called from Fortran programs with an efficient procedure for storing screen contents. The manual provides instructions for using the toolkit to build an effective interface to models or data bases. In doing so, the manual draws on an expanding body of knowledge of the "human factors" involved in effective design of man-machine interfaces. By providing a powerful set of "tools" and a detailed consideration of the best way to use these tools, the ANNIE-IDE software and manual are intended to greatly reduce the difficulty a programmer will encounter in developing an interactive user interface.

1.1 WHO SHOULD USE ANNIE-IDE?

The ANNIE-IDE package can be utilized to build a user interface for virtually any type of computer program or computer-stored data base. Potential applications range from generating user-supplied input for already existing scientific or environmental models to allowing user access to data bases for the purpose of searching, analyzing, or displaying data values. ANNIE-IDE provides application programmers with the tools needed to build and store a convenient user interface whenever and wherever there is a need to perform one or more of the following operations.

- display text
- prompt the user to select a menu option
- prompt the user to input or edit values in a one- or two-dimensional array
- open a file to store or retrieve text and other information

1.2 WHY USE ANNIE-IDE?

The advantages gained by using ANNIE-IDE for interface development center on four concepts: modularity, consistency, portability, and ease of application. The system utilizes a library of subroutines to perform the basic communication functions between the user and the program. Repeated use of these subroutines allows the application programmer to focus attention on designing an effective sequence for interactions while minimizing the effort needed to program the more mechanical aspects of the interface. At the same time the repeated use of modular subroutines decreases the likelihood of introducing programming errors. By approaching the interface development effort as a repetitive application of a limited number of tools, a higher degree of programming consistency is maintained both within and between resulting interactive programs; both the programmer and the user benefit from program consistency. Because the ANNIE-IDE software utilizes strict coding conventions, the machine dependency of interactive programs generated by its use is kept to a minimum, resulting in a high degree of portability to various hardware. The residual machine dependencies are identified in Section 2.4. Finally, the ANNIE-IDE system can make the task of developing a user interface much easier for an application programmer. The expediency of using ANNIE-IDE will become more evident in the remainder of this manual, particularly Sections 4 and 5.

1.3 MANUAL CONTENTS

Section 2 of this manual provides background on the development of the ANNIE-IDE system; an overview of the toolkit contents and application procedures; and a description of the system's scope, design requirements, goals, and limitations. Section 3 describes the theoretical considerations of interactive system design in general terms and then relates these considerations to the ANNIE-IDE package. In Section 4, general instructions are given for building and storing screen definition files, using the individual toolkit subroutines, and implementing a Fortran program that calls the subroutines and sequences user interactions. In Section 5, detailed examples of the application of the toolkit are provided. In Section 6, ideas for additional development activities are presented. In addition to the instructional materials in the main body of this manual, two appendices are provided. Appendix A is a glossary of technical terms used in the manual, and Appendix B is a programmer's reference manual that provides details related to toolkit subroutines, data structures, screen definition files, the formats for the screen definition file used to build an interactive program, specification of system parameters, and contents of the ANNIE-IDE distribution disks. Appendix B is designed for easy reference during the actual interface building activities. The ANNIE-IDE distribution disks include additional documentation for the toolkit subroutines.

SECTION 2

OVERVIEW

2.1 BACKGROUND

2.1.1 WDM and ANNIE

Over the last several years, the U.S. Environmental Protection Agency (USEPA) and the U.S. Geological Survey (USGS) have been developing an integrated framework for environmental models called the Watershed Data Management (WDM) System (Lumb and Kittle, 1986). This framework has been developed to facilitate the joint, integrated use of environmental models and relevant databases required for model application. The framework has been oriented towards computer models with intensive data requirements and contains a user interface named ANNIE (Lumb et al., 1988). ANNIE is organized as a toolkit of Fortran subroutines that allow manipulation of databases and screen information, and perform many of the functions required to support model operation. ANNIE helps users interactively store, retrieve, list, check, update, and analyze spatial, parametric, and time-series data for hydrologic models. A binary, direct-access file is used to store the data in a logical, well-defined structure called the WDM file. Many hydrologic and water-quality models and analysis tools developed by the USGS and the USEPA currently use the WDM file, ANNIE, or both.

ANNIE/WDM provides the user with a common data base for many applications, hence eliminating the need to reformat data from one application to another. Furthermore, the ANNIE/WDM system offers its users an expanding library of subroutines for graphics, user queries, graphic and data storage, and retrievals to enable a programmer to efficiently create software for highly specialized applications. Figure 2.1 provides a summary of ANNIE capabilities.

2.1.2 EMIFE

A second user interface toolkit, the Environmental Model Input File Editor (EMIFE) (General Sciences Corporation, 1987), has been developed recently as a result of a joint project between EPA's Office of Research and Development and Office of Toxic Substances. EMIFE is based on user interaction capabilities developed for the Graphical Exposure Modeling System (GEMS) and the microcomputer

Data Management

- Create space on a disk for a WDM file
- Define attributes for each data set that will be stored in the WDM file
- Add, modify, or delete time-series data from data sets
- Copy all or part of the data in one data set to another data set within the same WDM file
- Update the WDM file by deleting or renumbering data sets
- Export data sets from a WDM file to a flat file
- Import data sets from a flat file to a WDM file
- Load U.S. Geological Survey (USGS) and Hydrological Simulation Program-Fortran (HSPF) sequential files into WDM files
- Perform selected numerical transformations on one or two time series to produce a new time series

Data Display

- Display data-set attributes in list or tabular format
- List time-series data in a data set that are within a specified time span and value range
- List data in a data set that are within a specified time span but outside of a value range
- Display a summary of data sets for a WDM file
- Display a list of data sets that exhibit user-specified attributes
- Table a year of time-series data by month and day

Graphics

- Plot one or more time series over a specified time span
- Produce x-y plots for time-matched values from two data sets in a WDM file
- Produce x-y plots for time-matched values of two data types input by a terminal session
- Produce x-y plots of attribute pairs for all data sets or a selected subset of the data sets in a WDM file
- Produce a graphical representation of "the percent of time data values in a time series exceed specified levels" as determined by ANNIE statistical analysis

Analysis

- Perform flow-duration analysis (use all data in a time series)
- Compute error analysis statistics (use all data in two time series as a basis for computing absolute errors, standard errors, error matrix)
- Generate event statistics for a user-specified subset of a time series ("event" may be defined by value threshold or time span)
- Perform event frequency analysis according to generalized procedures or Water Resources Council (WRC) guidelines for determining flood flow frequency (USGS program J407, Bulletin 17B)

Model Input Preparation

- Prompt user for parameter values and options
- Check user entries against acceptable range of values
- Assemble input sequences
- Modify input sequence files

Figure 2.1. ANNIE functions and capabilities.

version PCGEMS (General Services Corporation, 1988), which were designed primarily to deal with models and data needed to evaluate fate and exposure of chemicals covered by the Toxic Substances Control Act. As part of the re-design effort accomplished in the EMIFE system, computer dependencies of the code were removed, and standard Fortran 77 was implemented. EMIFE is intended for use by applications programmers to develop easy-to-use interfaces to the environmental databases and models being developed by both the EPA and its contractors. This toolkit complements the original ANNIE user interface by adding additional input screen formats, but uses a less sophisticated scheme for management of the potentially large number of input screens required in a data intensive model. EMIFE capabilities allow full screen editing of three classes of input menu screens: a menu of input options, a menu of single value items, and a menu of one-dimensional arrays of variable sizes.

2.1.3 ANNIE-IDE

During the first half of 1988, the ANNIE and EMIFE capabilities were integrated into a consistent user interaction framework called the ANNIE Interaction Development Environment (ANNIE-IDE). ANNIE-IDE includes (1) a toolkit of subroutines that may be used by the programmer to build a standardized user interaction, (2) a means for storing and updating screen contents, and (3) a methodology for interactive software development. This manual will guide programmers, through explanation and example, in effective use of ANNIE-IDE.

2.2 OVERVIEW OF TOOLKIT AND METHODOLOGY

The ANNIE Interaction Development Environment (ANNIE-IDE) was created to provide a consistent methodology for building interactive interfaces for environmental computer programs and data bases. The ANNIE-IDE software combines a toolkit of subroutines for interface construction that can be called from Fortran programs with a convenient means for storing and updating screen contents. This manual provides instructions for using the toolkit to build an effective interface to both models and data bases. By providing a powerful set of tools and a detailed consideration of the best way to use these tools, the ANNIE-IDE software and manual can greatly reduce the difficulty an application programmer will encounter in developing an interactive interface.

The ANNIE-IDE toolkit consists of software that allows construction of screens, control of screen display, and performance of related housekeeping. Figure 2.2 illustrates the tasks that can be accomplished using the toolkit.

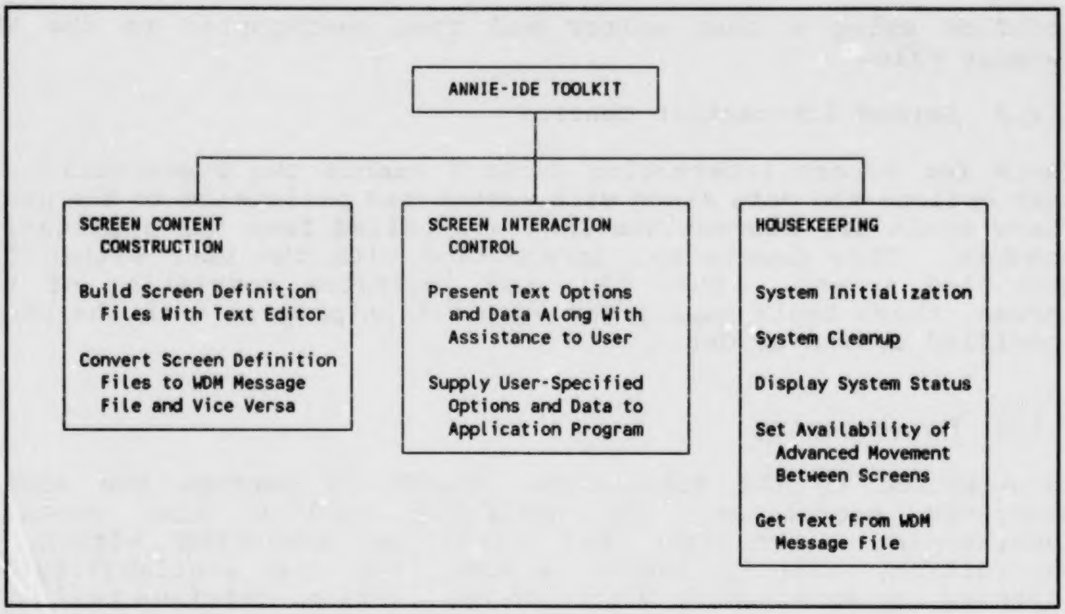

ANNIE-IDE TOOLKIT
SCREEN CONTENT CONSTRUCTION
Build Screen Definition Files With Text Editor
Convert Screen Definition Files to WDM Message File and Vice Versa
SCREEN INTERACTION CONTROL
Present Text Options and Data Along With Assistance to User
Supply User-Specified Options and Data to Application Program
HOUSEKEEPING
System Initialization
System Cleanup
Display System Status
Set Availability of Advanced Movement Between Screens
Get Text From WDM Message File

Figure 2.2. ANNIE-IDE toolkit.

2.2.1 Screen Content Construction

The content of screens is defined by using **screen definition directives**. The concept of "directives" was developed in ANNIE-IDE to (1) eliminate the code required to construct screens from the main body of the interactive program and (2) enable the application programmer to use consistent methods for screen construction. The directives provide the necessary information to the toolkit subroutines to control text and data display on the screens.

The contents of an individual screen are defined by a series of directives called a **screen group**. Screen groups for all screens needed by the interactive program are constructed in a sequential file by using a text editor. After this file has been developed, ANNIE-IDE provides a stand-alone interactive program called MESSIE that is used to convert the sequential file to a **WDM message file.** The WDM file is direct access and hence allows much-improved application program communication with large amounts of screen information. A single WDM message file can contain the contents of many individual screen definition files. In fact, more than three million unique screens could conceivably be stored in one WDM file. Use of the ANNIE-IDE software enables efficient modification of existing screens and addition of new screens. To accomplish these tasks, the MESSIE program is used to export the existing WDM message file contents back to a sequential file, which can be modified using a text editor and then re-imported to the WDM message file.

2.2.2 Screen Interaction Control

Tools for screen interaction control enable the presentation of text options and data along with associated assistance to the user. These tools are subroutines that are called from the application program. They handle all interaction with the user within the specified screen. When the user indicates completion of the screen, these tools supply the application program with the user-specified option or data.

2.2.3 Housekeeping

In addition to the subroutines needed to perform the above-mentioned operations, the ANNIE-IDE toolkit also contains housekeeping subroutines to initialize ANNIE-IDE within an application, display system status, set the availability of advanced screen movement within an application, retrieve text from the WDM message file and shut down ANNIE-IDE upon completion of its use. Full instructions and examples for using the system for these purposes are provided in Section 5.

2.2.4 Sample Screens

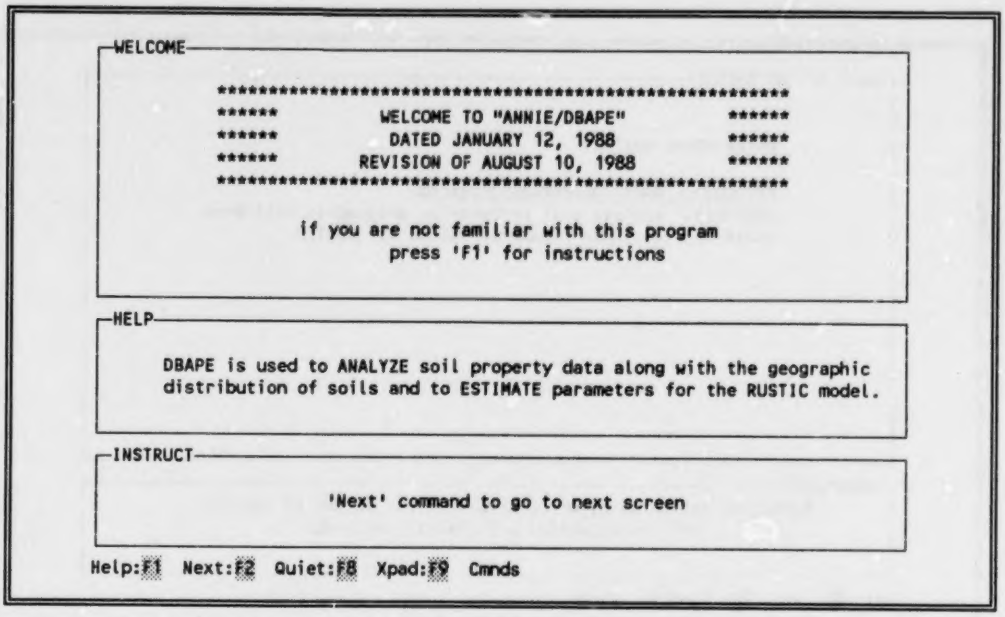
The subroutines for screen display control are divided into five groups by function:

- display text
- select an option from a menu
- evaluate a one-dimensional (1-D) array of related values
- evaluate a two-dimensional (2-D) array of related values
- open a file to store or retrieve text and other information

The overall function of the toolkit subroutines can best be illustrated by providing sample screens (Figures 2.3 - 2.7) for each of these five functions which have been generated by using the ANNIE-IDE software.

The TEXT screen (Figure 2.3) is used to display information to the user. It may contain up to 50 lines of text, with 10 or 16 available for viewing at any time. The user may use the cursor movement keys to view other portions of the text. The 'NEXT' command is used to move on to the next screen.

The MENU screen (Figure 2.4) is used to select an option. The selection may be made using the first letter of the option(more letters if a conflict exists) or the cursor movement keys and the "NEXT" command.



```
-WELCOME-----------------------------------------------------------

          ******************************************************
          ******        WELCOME TO "ANNIE/DBAPE"          ******
          ******         DATED JANUARY 12, 1988           ******
          ******      REVISION OF AUGUST 10, 1988         ******
          ******************************************************

                  if you are not familiar with this program
                        press 'F1' for instructions

-HELP--------------------------------------------------------------

     DBAPE is used to ANALYZE soil property data along with the geographic
     distribution of soils and to ESTIMATE parameters for the RUSTIC model.

-INSTRUCT----------------------------------------------------------

                  'Next' command to go to next screen

Help:F1  Next:F2  Quiet:F8  Xpad:F9  Cmnds
```

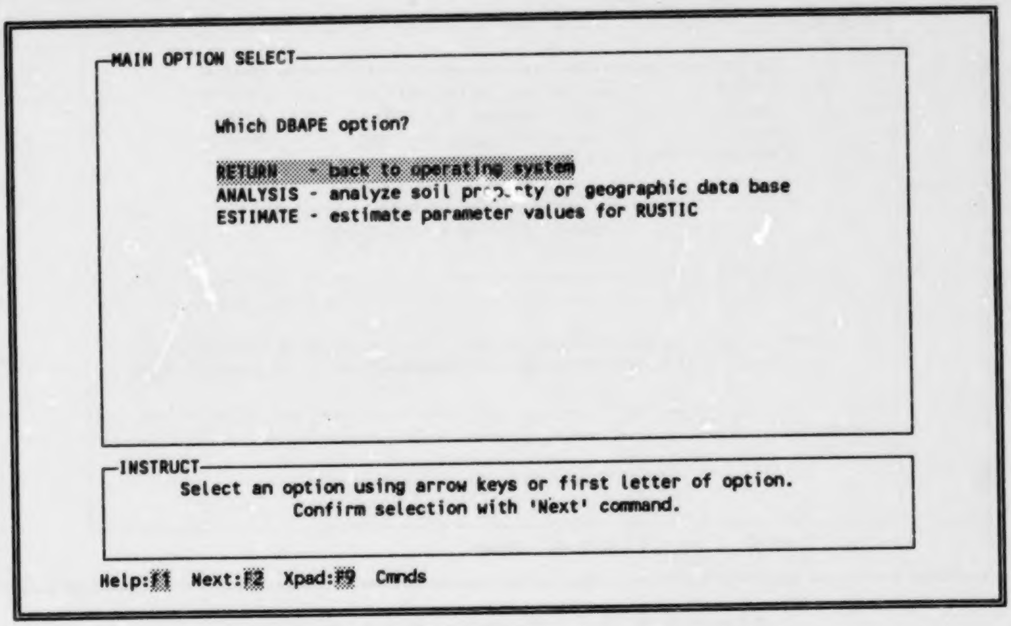
Figure 2.3. Sample text screen.

The 1-D ARRAY screen (Figure 2.5) is used to enter numeric values or text. Up to 24 fields of information may be entered. The user may use the cursor keys to move between fields. Details about each field may be requested with the "HELP" and "LIMITS" commands. User input is checked against valid values and errors are reported if incorrect input is detected. The user may adjust the value for the currently highlighted field by specifying an arithmetic expression (e.g., "*3".) The "NEXT" command is used to go to the next screen.

The 2-D ARRAY screen (Figure 2.6) is used to enter arrays of numeric values or text. Up to 24 fields and 49 rows of information may be entered. Movement within this screen is similar to that in the 1-D ARRAY screen with the additional feature of "WINDOW." This allows manipulation of large blocks of data simultaneously.

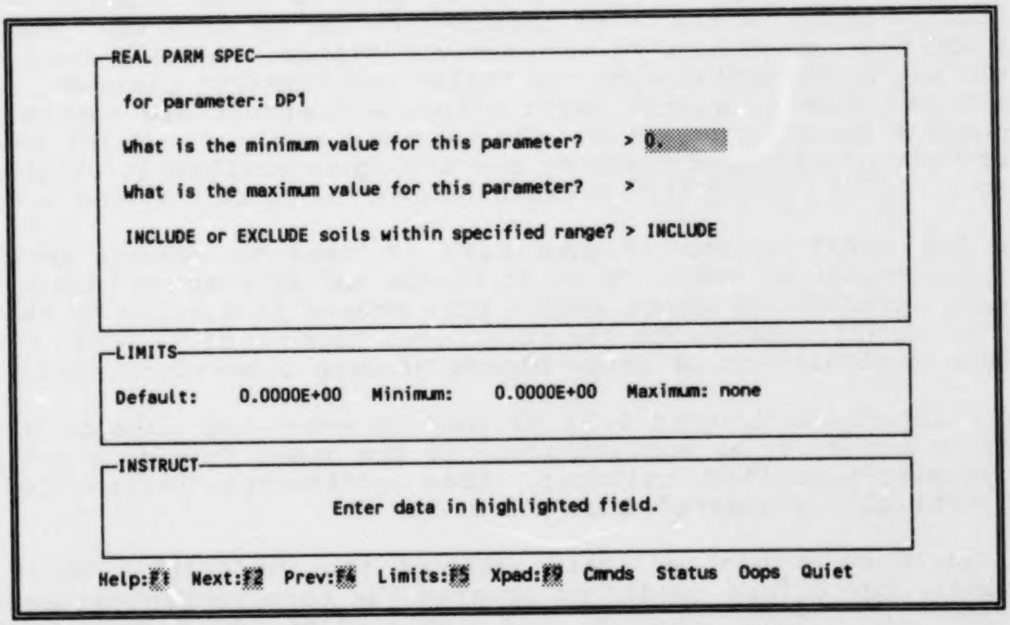
The FILE screen (Figure 2.7) is used to enter the name of a file that is needed by an application. If the named file does not meet programmer-specified criteria, then additional information is automatically requested from the user.

It should be emphasized that, although the ANNIE-IDE toolkit can minimize the effort needed to program the more mechanical aspects of an interactive program, the application programmer remains responsible for designing an effective sequence for user



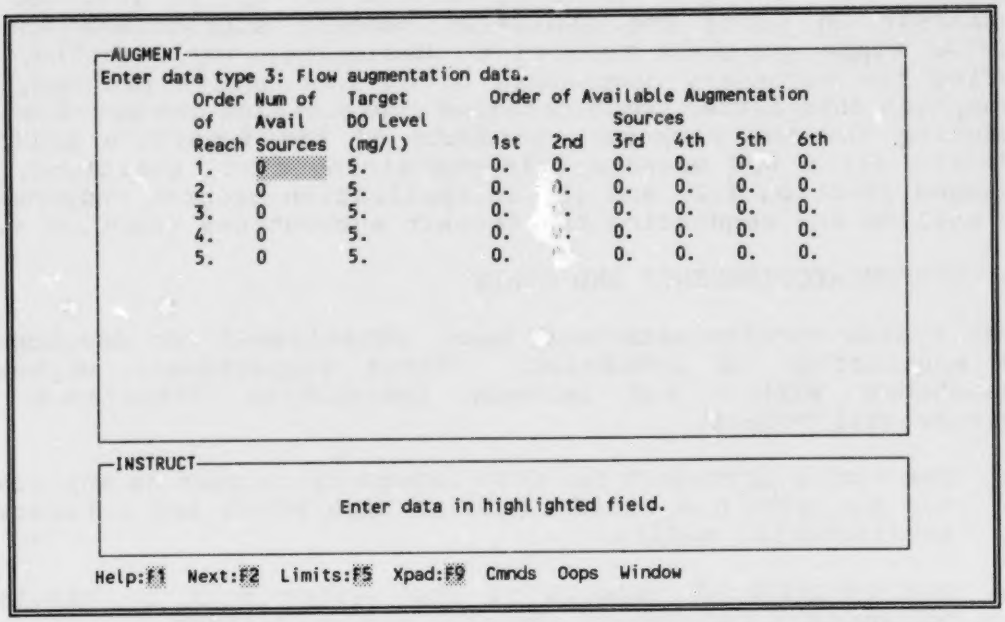
```
MAIN OPTION SELECT

        Which DBAPE option?

        RETURN   - back to operating system
        ANALYSIS - analyze soil property or geographic data base
        ESTIMATE - estimate parameter values for RUSTIC

INSTRUCT
      Select an option using arrow keys or first letter of option.
                 Confirm selection with 'Next' command.

Help:F1  Next:F2  Xpad:F9  Cmnds
```

Figure 2.4. Sample menu screen.



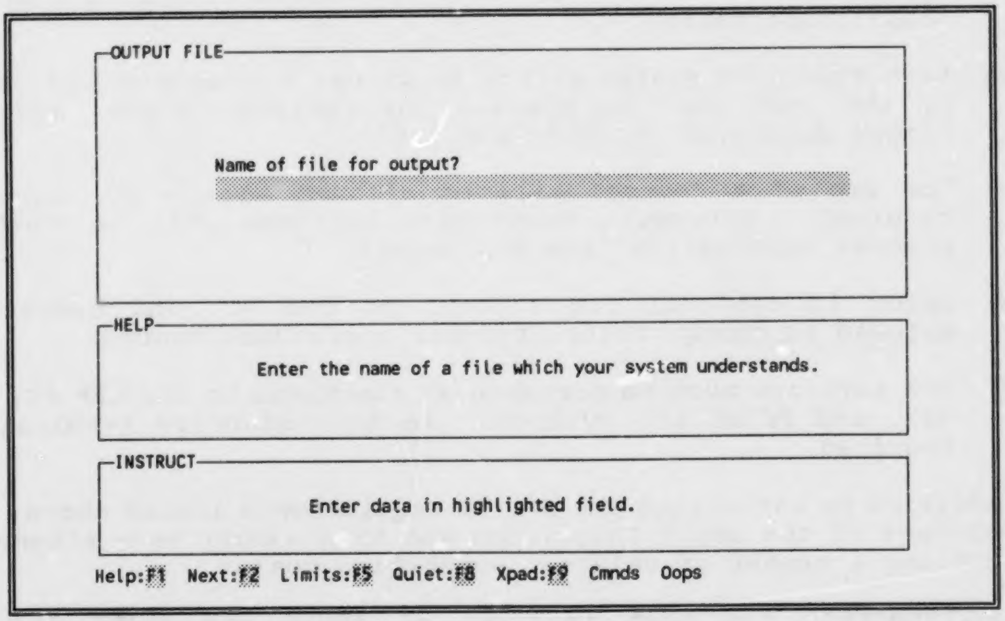
```
REAL PARM SPEC

 for parameter: DP1

 What is the minimum value for this parameter?    > 0.

 What is the maximum value for this parameter?    >

 INCLUDE or EXCLUDE soils within specified range? > INCLUDE

LIMITS
Default:    0.0000E+00  Minimum:    0.0000E+00  Maximum: none

INSTRUCT
                 Enter data in highlighted field.

Help:F1  Next:F2  Prev:F4  Limits:F5  Xpad:F9  Cmnds  Status  Oops  Quiet
```

Figure 2.5. Sample 1-D array screen.



```
AUGMENT
Enter data type 3: Flow augmentation data.
            Order Num of  Target          Order of Available Augmentation
            of    Avail   DO Level                     Sources
            Reach Sources (mg/l)          1st   2nd   3rd   4th   5th   6th
            1.    0       5.              0.    0.    0.    0.    0.    0.
            2.    0       5.              0.    0.    0.    0.    0.    0.
            3.    0       5.              0.    0.    0.    0.    0.    0.
            4.    0       5.              0.    0.    0.    0.    0.    0.
            5.    0       5.              0.    0.    0.    0.    0.    0.

INSTRUCT
                  Enter data in highlighted field.

Help:F1  Next:F2  Limits:F5  Xpad:F9  Cmnds  Oops  Window
```

Figure 2.6. Sample 2-D array menu screen.



```
OUTPUT FILE

           Name of file for output?

HELP
            Enter the name of a file which your system understands.

INSTRUCT
                  Enter data in highlighted field.

Help:F1  Next:F2  Limits:F5  Quiet:F8  Xpad:F9  Cmnds  Oops
```

Figure 2.7. Sample file screen.

interactions. The dual goals of this manual are to instruct the programmer on using the ANNIE-IDE toolkit subroutines and to provide ample guidance related to designing, implementing, and storing the necessary components of an interactive program. To accomplish this latter goal, detailed instructions are provided for producing the two necessary products of the interface building effort: (1) a WDM message file containing text, questions, and messages (Section 4.2) and (2) an application program responsible for calling and sequencing the toolkit subroutines (Section 4.3).

2.3 SYSTEM REQUIREMENTS AND GOALS

Seven system requirements have been established for development and application of ANNIE-IDE. These requirements emphasize consistency within and between interactive interfaces for environmental models.

(1) The system framework for user interactions must be applicable for use within a broad range of both batch and interactive environmental models.

(2) The operational aspects of the system must be developed following a consistent approach and guidelines.

(3) User interactions and procedures must be consistent irrespective of the model/database to which the toolkit is applied. For example, the same key should always be used for 'HELP' information.

(4) Both model and system parameter values and options specified by the user must be checked for correct ranges, and the ranges displayed whenever available.

(5) The use of a 'mouse' will be allowed on the PC, but not required. Microsoft Mouse Menu routines will be used to process information from the mouse.

(6) Color is optional for screens on the PC, and users are allowed to change colors to suit individual tastes.

(7) The software must be portable at a minimum to the IBM PC, DEC VAX, and PRIME 850 systems. An ANSI standard terminal is required.

In addition to satisfying the system requirements listed above, the developers of the ANNIE-IDE system and this manual have attempted to achieve a number of related, supporting goals:

(1) Integrate the best features of ANNIE and EMIFE into a consistent user interaction framework.

interactions. The dual goals of this manual are to instruct the programmer on using the ANNIE-IDE toolkit subroutines and to provide ample guidance related to designing, implementing, and storing the necessary components of an interactive program. To accomplish this latter goal, detailed instructions are provided for producing the two necessary products of the interface building effort: (1) a WDM message file containing text, questions, and messages (Section 4.2) and (2) an application program responsible for calling and sequencing the toolkit subroutines (Section 4.3).

2.3 SYSTEM REQUIREMENTS AND GOALS

Seven system requirements have been established for development and application of ANNIE-IDE. These requirements emphasize consistency within and between interactive interfaces for environmental models.

(1) The system framework for user interactions must be applicable for use within a broad range of both batch and interactive environmental models.

(2) The operational aspects of the system must be developed following a consistent approach and guidelines.

(3) User interactions and procedures must be consistent irrespective of the model/database to which the toolkit is applied. For example, the same key should always be used for 'HELP' information.

(4) Both model and system parameter values and options specified by the user must be checked for correct ranges, and the ranges displayed whenever available.

(5) The use of a 'mouse' will be allowed on the PC, but not required. Microsoft Mouse Menu routines will be used to process information from the mouse.

(6) Color is optional for screens on the PC, and users are allowed to change colors to suit individual tastes.

(7) The software must be portable at a minimum to the IBM PC, DEC VAX, and PRIME 850 systems. An ANSI standard terminal is required.

In addition to satisfying the system requirements listed above, the developers of the ANNIE-IDE system and this manual have attempted to achieve a number of related, supporting goals:

(1) Integrate the best features of ANNIE and EMIFE into a consistent user interaction framework.

(2) Enable a system for storing and updating user interaction text and parameter ranges.

(3) Incorporate effective error handling methods.

(4) Reduce memory requirements for interactive systems.

(5) Improve user control over interaction sequencing.

(6) Provide system design guidance based on "human factors" research.

2.4 LIMITATIONS

ANNIE-IDE and its manual are designed to aid the application programmer in several critical components of interactive program development. However, the limit of their scope and utility should be kept in mind from the onset of a planned application. Consequently, several key functional limitations of the ANNIE-IDE software and manual are outlined below:

(1) The current ANNIE-IDE is limited to a large extent to input functions. Data management and output functions (e.g., plotting) are beyond the scope of the current project and hence the programmer must develop such functions parallel to, but without the assistance of, ANNIE-IDE. (Provision has been left for plotting within the two-dimensional array screen, but has not been implemented.)

(2) The programmer is responsible for all file formats and data structures required by the specific model for which an interactive interface is being developed.

(3) The toolkit subroutines and manual instructions emphasize "intrascreen functions." Although some aid is provided for implementing "interscreen" communication, these functions are outside the scope of work of the ANNIE-IDE development project and hence have not been fully developed.

In addition to these functional limitations, ANNIE-IDE also is subject to a small number of hardware portability limitations that result from the use of extensions to Fortran as listed below. All these extensions are supported on the IBM PC, the Dec VAX, and the Prime 850. If the planned interactive code is to be used on other hardware, it is the programmer's responsibility to assure that the following extensions are allowable on the target hardware:

(1) Reading single characters without a carriage return

(2) Writing a string of characters without a carriage return

(3) Using ANSI standard terminal cursor movement for the following functions: "home cursor," "cursor position," "cursor relative move," and "clear screen."

2.5 TOOLKIT DEMONSTRATION EXAMPLES

As a component of the ANNIE-IDE development project, the toolkit has been used to develop an enhanced user interaction for an already-existing environmental model, QUAL2E-UNCAS (Brown and Barnwell, 1987), and for a recently developed soils properties analysis program, DBAPE (unpublished). Because the examples provided in this manual are all excerpted from the development of the user interaction for QUAL2E-UNCAS and DBAPE, a brief description of each is provided for programmers who are unfamiliar with them.

QUAL2E-UNCAS and its predecessors QUAL-II and QUAL-2E are probably the most widely used computer models for predicting the effects of conventional pollutants in streams. The model simulates the following water quality constituents: dissolved oxygen, biochemical oxygen demand, temperature, algae, nitrogen and phosphorus species, coliforms, an arbitrary nonconservative constituent, and three conservative constituents. QUAL2E-UNCAS is a steady-state hydraulics model with both steady-state and quasi-dynamic water quality capabilities. Many model parameter values required by this program are variable by reach, and certain meteorologic parameters are variable with time. QUAL2E-UNCAS also allows the modeler to perform uncertainty analyses for specified model parameters by utilizing Monte Carlo simulation techniques.

QUAL2E-UNCAS was selected for demonstration of the ANNIE-IDE capabilities for several reasons. First, the model has evolved from predecessor models that have been effectively used for analysis of numerous environmental problems by a wide user audience. Second, the model already had an interactive interface developed through the use of ANNIE. This fact allowed the development of enhanced interactive capabilities for a rather complex model with a lesser level of effort than would have otherwise been required. Because the model requires considerable input, and input of many different kinds, application of ANNIE-IDE to QUAL2E-UNCAS was a good test of the toolkit's ability to provide the programmer with the capabilities needed to build an interactive interface for any environmental model.

The Data Base Analyzer and Parameter Estimator (DBAPE) is an interactive program currently under development at the Environmental Research Laboratory, Athens, Georgia. At present DBAPE interacts with two data bases: the first contains a compilation of the acreage of individual agricultural soils types (by USDA soil classification) across the United States on a county basis; the second is a compilation of soils characteristics for each USDA soil classification such as crop suitability, average depth of soil horizons, percent sand, silt, and clay, SCS runoff classification (hydrologic group), and other pertinent physical data. The current version of DBAPE allows the user to locate and rank the acreage of a particular soil type by county, find all the soil types which exhibit one or more of the characteristics compiled in the data base (e.g., soils classified as belonging to SCS hydrologic group B and also suitable for growing corn), and export the results of any interactive operation to a file or the line printer.

Examples from both QUAL2E-UNCAS and DBAPE are used in Section 5 to illustrate general procedures involved in an ANNIE-IDE application. The source code and file definition directives for these two applications are found on the distribution disks for ANNIE-IDE.

SECTION 3

THEORETICAL CONSIDERATIONS OF ANNIE-IDE DEVELOPMENT

Whereas subsequent sections of the manual emphasize the mechanical aspects of utilizing ANNIE-IDE to facilitate the development of interactive programs, this section describes the theoretical considerations of interactive system design and relates these considerations to the design and anticipated programmer applications of the ANNIE-IDE package. The discussion presented in this section relies heavily on the results of current "human factors" research which is providing a much better understanding of the requirements for effective man-machine interfaces.

3.1 GENERAL THEORY OF INTERACTIVE INTERFACES

The purpose of this section is to expose programmers who are undertaking an ANNIE-IDE application to the general principles that should govern the development of an interactive program interface. The ideas and guidelines presented here are products of the expanding field of human factors research. Recently, a number of books and numerous research efforts have been devoted to defining improved strategies for effective human-computer interaction. Although a comprehensive consideration of human factors principles is beyond the scope of this manual, a basic understanding of these principles is necessary to appreciate the motivation for ANNIE-IDE design and application instructions.

The material presented here originates from two sources: the notes and comments of Professor Albert Badre who served as a technical consultant to the ANNIE-IDE development effort and a book entitled Designing the User Interface by Professor Ben Shneiderman (1987). Topics addressed include:

- human factors goals
- why interactive systems fail
- rules of system design
- interactive system styles
- menus
- form fill-in

The intent of presenting this material is twofold: first, to explain the foundation on which the ANNIE-IDE toolkit and application instructions are based, and second, to provide guidelines for developing those components of the interactive program that are not direct products of applying ANNIE-IDE. Hence, although all the material contained in this section will prove useful in developing an interactive program, only a portion, as identified in later portions of this section, is embodied in the current ANNIE-IDE system. Without question, application of the general guidelines outlined below will improve the quality of all aspects of an interactive program; consequently, the application programmer is encouraged to review this section of the manual on an regular basis during the program development effort.

3.1.1 Human Factors Goals

It is generally accepted that the ultimate goal of software is "usability." Usability is achieved by: (1) creating an environment in which the user feels in control; that is, the user "drives" the system in the same sense that a driver controls a car; and (2) reducing the level of effort needed by the end-user to complete tasks. Hence, the overall goals of interface development should be to:

- minimize time required to learn, search for information, acquire knowledge, or understand the computer task to be performed
- maximize speed of performance of tasks
- reduce user errors
- provide the user with protection from danger of system failure, thereby encouraging exploration of system capabilities
- maximize satisfaction of users

Human factors researchers recognize that there are necessary tradeoffs between these goals; consequently, priorities must be identified and ranked at the onset of each interface development project. Shneiderman (1987) notes that success in achieving all the above listed goals can be quantified; hence, usability of interactive software need not be a subjective term.

3.1.2 Why Interactive Systems Fail

Interactive systems that are deemed "failures" by the user community are often hindered by the following inadequacies:

- inconsistent internal style and methods
- inadequate on-line user assistance
- rigid sequencing that does not allow the user to efficiently perform frequent tasks
- program operations that cause the user to feel a loss of control

3.1.3 Rules of System Design

Shneiderman presents the following list of underlying principles for design of interactive systems.

- strive for consistency
- provide shortcuts for frequent users
- offer informative feedback
- design dialogues that indicate the completion of operations
- offer simple error handling
- permit easy reversal of actions
- allow users to initiate actions rather than to solely respond to program queries
- minimize the user's short-term memory load

3.1.4 Styles for Interactive Systems

In constructing an interactive program, the application programmer may utilize a variety of interaction styles. The scope of work for development of the ANNIE-IDE system was restricted to functions related to menu selection and form fill-in; hence, the remainder of this section is devoted to outlining general guidance related only to these two styles of interaction. For many applications, a combination of the two interaction styles is appropriate. The advantages and disadvantages of using these styles are summarized in Table 3.1.

TABLE 3.1. ADVANTAGES AND DISADVANTAGES OF CANDIDATE INTERACTION STYLES (ADAPTED FROM SHNEIDERMAN (1987))

Interaction Style	Advantages	Disadvantages
Menu selection	shortens learning reduces keystrokes structures decision-making permits use of dialogue management tools easy to support error handling	danger of many menus may slow frequent users consumes screen space requires rapid display rate
Form fill-in (screens)	simplifies data entry requires modest training assistance is convenient permits use of form management tools	consumes screen space

3.1.5 Menu Selection Interactions

Effective menu selection systems are the result of careful consideration and testing of numerous design issues such as menu organization and structure, the number and sequence of menu items, titling, prompting format, graphic layout and design, phrasing of menu items, display rates, response time, shortcuts through the menus for knowledgeable frequent users, availability of help, and the selection mechanism (keyboard, pointing devices, etc.). Guidance that will be particularly useful for developing an interface using ANNIE-IDE is summarized below. Included are rules for grouping in menus and menu tree structures, rules for sequencing the selections in a menu, and general considerations for menu construction. If the programmer desires more detailed guidance on any of the items presented, reference to Shneiderman (1987) or other currently available human factors texts is recommended.

3.1.6 Rules for Grouping Items in Menus and Tree Structures

Adhering to the following rules will result in an effective grouping of menu items for user interaction:

- limit menus to seven items or less
- create groups of logically similar items
- form groups that cover all possibilities
- make sure items are nonoverlapping
- use familiar terminology, but ensure items are distinctive from each other

3.1.7 Menu Item Presentation Sequence

If items have a natural sequence, ordering should reflect it; examples are:

- time
- number
- physical properties

For items with no apparent natural sequence, ordering may be according to:

- alphabetic sequence of terms
- grouping of related items
- most frequently used items first
- most important items first

3.1.8 General Considerations (adapted from Shneiderman (1987))

- describe menu items briefly and in a consistent grammatical style
- use consistent layout and terminology
- give position in organization by graphic layout, numbering and titles
- use menu items as menu titles for menus of next lower level
- permit type-ahead, jump-ahead, and other short-cuts
- permit jumps to previous and main menu
- offer "help" facilities

3.1.9 Form Fill-in Interactions

If data entry of names and numeric values is needed, and especially when many fields of data are necessary, the appropriate interaction style is form fill-in. General guidelines for developing interactive screens for form fill-in, as suggested by Shneiderman (1987), are:

- give the form a meaningful title
- use comprehensible instructions
- use logical grouping and sequencing of fields
- develop a visually appealing layout
- use familiar field labels
- use consistent terminology and abbreviations
- provide error correction for characters and fields
- utilize visual templates for common fields
- implement "help" facilities

A final general comment concerning interface development should be of considerable interest to application programmers planning to use ANNIE-IDE to build an interactive interface. In his book Designing the User Interface, Shneiderman singles out computer use in exploratory environments, such as environmental modeling, as the most difficult in which to develop successful interfaces. He attributes this difficulty to three problems:

- user tasks tend to be less repetitive and require more system flexibility
- system usage tends to be less frequent, increasing the need for on-line assistance and good retention characteristics
- users tend to have high expectations of the systems they apply

Using the ANNIE-IDE toolkit during interactive program development cannot free the programmer from dealing with any of these inherent problems. It can, however, allow the programmer to minimize the effort needed to develop the mechanical aspects of the interface and hence focus attention on the more challenging components of interface development intrinsic to the domain of environmental science and engineering.

3.2 APPLICATION OF HUMAN FACTORS THEORY TO ANNIE-IDE

To better understand how the theoretical considerations of human factors research are embodied in ANNIE-IDE, it is useful to differentiate between "intrascreen" and "interscreen" functions in an interactive program. The focus of the ANNIE-IDE toolkit is to provide assistance in developing intrascreen functions; that is to say, the tools allow the programmer to build individual screens. Hence, in the strictest sense, a discussion of the theoretical basis for ANNIE-IDS could be restricted to the realm of screen building. It has been a goal of this project, however, to provide sufficient instructions and examples in the manual to guide application programmers in the construction of a complete interactive program. Thus, guidance also has been given on interscreen functions such as control logic and screen sequencing. ANNIE-IDE is a toolkit of utility subroutines and instructions for developing and storing two parallel products: a file containing text, questions, and messages used in interactive communication, and a Fortran program containing the control strategy and sequencing instructions for interactions. Consequently, the discussion of theoretical considerations for ANNIE-IDE will not only consider the basis of the ANNIE-IDE software, but its current and recommended application procedures as well.

Goals for enhancing human factors focus on program control, reliability, functionality, and consistency. Concern for all these goals is reflected in the design of the ANNIE-IDE toolkit and in its first two applications to the programs QUAL2E-UNCAS and DBAPE. The toolkit itself is designed to allow the application programmer to utilize the capabilities of over 200 utility subroutines while dealing directly with approximately 40 lead subroutines. Program functions can thus be performed reliably and consistently by repeated use of the necessary toolkit components. By using ANNIE-IDE, the programmer is encouraged to recognize and isolate data management functions, thereby avoiding unnecessary clutter and complexity in the interactive program code.

Additional manifestations of human factors theory are evident in the sample screens developed during the first two toolkit applications and used as illustrations throughout this manual.

Features of ANNIE-IDE that embody human factors theory include:

- o screen layout
- o data window
- o assist window
- o instruction window
- o command line

- user control within screens
- screen definition directives
- screen sequencing

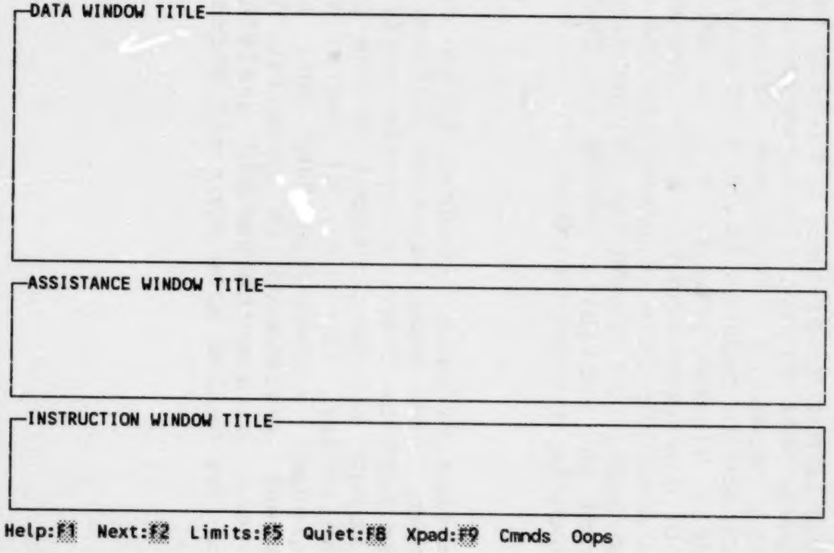
The following sections provide details on the use of human factors concepts in the implementation of ANNIE-IDE.

3.3 SCREEN LAYOUT

The impetus for developing ANNIE-IDE is recognition that a need exists for an straightforward, consistent methodology for building interactive interfaces. Although the focal point in developing ANNIE-IDE is to provide tools and guidance that will make the program development effort of the application programmer easier and more productive, we recognize that the ultimate success or failure of the effort depends on the user satisfaction with the final product. One of the most important determinants of user satisfaction is screen layout. If the "look and feel" of communication screens can be made consistent, user comfort and confidence in using the system is greatly increased.

To this end, we have made the general screen format resulting from applying ANNIE-IDE an intrinsic feature. That is to say, while the application programmer has a large degree of control over what information appears on the screen, and the user has control over when the information appears, the ANNIE-IDE software determines where the information appears on the screen. By making screen layout an intrinsic feature of ANNIE-IDE, the user always knows where to look for specific kinds of information on the screen, and the location remains the same for every screen in every interactive program produced using ANNIE-IDE.

Figure 3.1 defines the basic layout of an ANNIE-IDE screen. Screen information is divided into four components: three windows (data window, assistance window, instruction window) and the command line. For convenience, the dimensions, content, and important features of the four screen components are summarized along the periphery of the screen area in the figure. Additional details on types of content and features for each of these four screen components are provided in sections 3.4-3.7. General construction details for screen content are provided in Section 4 and illustrated in Section 5.



COMMAND LINE

DIMENSIONS : 80 X 1
CONTENT : MENU OF ABBREVIATIONS FOR AVAILABLE COMMANDS
FEATURES : COMMANDS ARE AUTOMATICALLY ORDERED BY EXPECTED FREQUENCY OF USE
STANDARD SET OF COMMANDS AVAILABLE FOR BASIC APPLICATION PROGRAMS (SEE TABLE 3.3)
PROGRAMMER MAY ADD COMMANDS FOR ADVANCED APPLICATION PROGRAMS (SEE TABLE 3.4)

DATA WINDOW

TITLE : ANY PROGRAMMER - SPECIFIED STRING (UP TO 48 CHARACTERS)
DIMENSIONS : 78 X 16 (WITHOUT ASSISTANCE INFO WINDOW)
78 X 10 (WITH ASSISTANCE INFO WINDOW)
CONTENT : INFORMATION PROVIDED BY APPLICATION PROGRAMMER THROUGH SCREEN DEFINITION FILE
FEATURES : ACCOMMODATES DATA FILE OF MAXIMUM DIMENSIONS 78 X 50 WITH SCROLLING

ASSISTANCE WINDOW

TITLE : HELP, COMMANDS, LIMITS, STATUS, OR XPAD
DIMENSIONS : 78 X 4
CONTENT : INFORMATION PROVIDED BY SYSTEM (COMMANDS), APPLICATION PROGRAMMER (HELP,LIMITS, STATUS), OR USER (XPAD)
HELP - DETAILED USER ASSISTANCE FOR SCREEN CONTENTS
COMMANDS - DEFINITIONS OF AVAILABLE COMMANDS
LIMITS - RANGE OF ACCEPTABLE VALUES
STATUS - SYSTEM STATUS INFORMATION
XPAD - USER NOTES AND REMINDERS
FEATURES : ALLOWS SCROLLING IF INFORMATION EXCEEDS VERTICAL DIMENSION

INSTRUCTION WINDOW

TITLE : INSTRUCT OR ERROR
DIMENSIONS : 78 X 3
CONTENT : SYSTEM GENERATED INSTRUCTIONS AND ERROR MESSAGES
FEATURES : DIRECTS USERS TO ACCEPTABLE KEY STROKES

Figure 3.1. Screen format utilized by ANNIE-IDE.

3.4 DATA WINDOW

The top portion of the screen is the **data window**. The data window contents consist of one or more of the following.

(1) programmer-supplied prompts for user-supplied decisions by means of menu selection (see data window entitled "MAIN OPTION SELECT" in Figure 2.4 for example)
(2) programmer-supplied prompts for user-supplied data by means of form fill-in (see data window entitled "AUGMENT" in Figure 2.6 for example)
(3) echoes for current state of data (see data window entitled "WELCOME" in Figure 2.3 for example)
(4) analysis results

The title for each data window is specified by the programmer during the screen definition process and can be up to 48 characters long. Program structure can be emphasized by using the menu options from the data window of a menu screen as the titles for the data windows of the subordinate screens.

ANNIE-IDE allows for two user-controlled "sizes" for the data window. The default dimensions are 78 X 16 characters; in this scenario the assistance window is not displayed, resulting in a two window, one command line screen (see Figure 2.3 for example). If the user desires any of the forms of assistance described in Section 3.5, then the data window is reduced to 78 X 10 characters to accommodate the assistance window (Figure 3.1). ANNIE-IDE currently accommodates data of dimensions up to 78 X 50 and enables scrolling in the data window when the data size exceeds the window size. If necessary, the programmer can expand the height dimension (50) by increasing the size of the variables ZMNTXT and ZMNTX1 in common block ZCNTL2 (see Appendix B.3 for details.)

3.5 ASSISTANCE WINDOW

Within an ANNIE-IDE application program, several types of user assistance can be implemented. Some assistance is automatically generated by ANNIE-IDE, some must be supplied by the application programmer, and a limited amount may be provided by the program user. For the sake of organization, the forms of assistance can be classified into two groups: screen-dependent and screen-independent. **Screen-dependent assistance** is specific to the contents of a particular screen; **screen-independent assistance** is more global in nature and can be called upon from all screens.

Conceptually, three layers, or levels, of assistance can be provided for screen-dependent information:

- Level 1- Use of descriptive and unique words or abbreviations for field or menu option names in the data window provides "first-cut" definitions.
- Level 2- When space allows, additional information in the data window near the data field or menu option can clarify the desired information.
- Level 3- If additional parameter- or screen-specific assistance is needed, it can be supplied in the assistance window. The two types of screen-dependent assistance allowable in the assistance window are HELP and LIMITS.

The rationale for providing a means for "layered" help should be explained. Perhaps the most important goal in designing user assistance for an interactive program is to provide the proper amount of assistance at the proper time. The layered interaction strategy of ANNIE-IDE is designed so that the user must specifically request the higher levels of assistance; consequently, experienced users are not subjected to unnecessary information.

Screen-independent, or global, assistance provides a fourth level of help in the ANNIE-IDE software. There are four distinctive types of screen-independent assistance. Two types (XPAD and CMNDS) are implicit to ANNIE-IDE, and the other two types (STATUS and TUTOR) must be implemented by the application programmer.

The **assistance window** is used to display level 3 screen-dependent assistance (HELP and LIMITS) and all types of screen-independent assistance (CMNDS, XPAD, TUTOR, and STATUS).

The function, source, and size limitations for each type of assistance are summarized in Table 3.2. The user selects one assistance type at a time and the available assistance of that type is displayed in a 78 X 4 window directly below the data window. The title of the window corresponds to the type of assistance which has been requested by the user. The types of assistance which are available for a screen are indicated by the options listed in the command line (Section 3.7). When the application programmer adds a new type of assistance for a screen (Section 3.10), ANNIE-IDE automatically adds a corresponding command to invoke the new type of assistance to the command line. However, the command line is limited to 80 characters. If the size of the available assistance exceeds the window size, ANNIE-IDE automatically enables scrolling in the window.

TABLE 3.2. ASSISTANCE WINDOW TYPES

ASSISTANCE TYPE	FUNCTION	SOURCE OF ASSISTANCE	ALLOWABLE SIZE
HELP	PROVIDES INFORMATION WHICH CLARIFIES PARAMETER DEFINITIONS AND PROGRAM QUERIES	PROGRAMMER	NO LIMIT
LIMITS	DISPLAYS ALLOWABLE VALUES FOR AN ENTRY	PROGRAMMER	NO LIMIT
CMNDS	DISPLAYS AND DEFINES AVAILABLE COMMANDS	ANNIE-IDE	NO LIMIT
TUTOR	PROVIDES DETAILED PROGRAM TUTORIAL	PROGRAMMER	NO LIMIT
STATUS	DISPLAYS PROGRAMMER-DESIGNED SYSTEM STATUS MESSAGES WHICH SUMMARIZE PREVIOUS ACTIONS AND INDICATES RELATIVE LOCATION OF USER WITHIN PROGRAM STRUCTURE	PROGRAMMER	10 LINES
XPAD	ALLOWS USER TO WRITE NOTES AND REMINDERS DURING INTERACTIVE SECTION	USER	10 LINES

Details on each of the assistance types follow.

HELP

HELP assistance provides further information on model and system parameters and menu options. As noted above, HELP text must be provided by the application programmer and is specific to a particular screen. There are no size limitations on the HELP text, and it can be scrolled in the assistance window.

LIMITS

LIMITS displays, in the assistance window, the allowable values for a specific field in the data window. LIMITS information may be maximum and minimum acceptable numeric values or a list of acceptable alphanumeric values. LIMITS text must be provided by the application programmer and is specific to the identified field in the data window.

There are no size limitations on LIMITS text and it can be scrolled in the assistance window.

CMNDS

CMNDS displays the names and definitions of all active commands at the current location within the interactive program. The command definitions are intrinsic to ANNIE-IDE; hence, the definitions never change. It should be noted, however, that the list of available commands varies according to location within the program. For example, the programmer may elect to use the STATUS command only at certain program "levels." The availability of commands also may be influenced by the contents of the data window of the current screen. For example, the WINDOW command (Section 3.7) is only available when two-dimensional data are present in the data window, and the scrolling commands (UPPG, DNPG) are only available when data for the current data window exceed the window dimensions.

There are no size limitations on CMNDS text, and it can be scrolled in the assistance window.

TUTOR

TUTOR provides a general tutorial for all aspects of the interactive program. A single body of text, which must be provided by the application programmer, is available at all times during the interactive session. Whenever TUTOR assistance is requested by the user, the entire tutorial text is called up, and the user must scroll through the text within the assistance window to search for the desired information. There are no size limitations on the TUTOR text.

STATUS

STATUS assistance displays programmer-designed system status messages that summarize previous actions and indicate the relative location of the user within the program structure. Incorporating comprehensive STATUS assistance into the interactive program can be the programmer's most effective means of assuring that the user does not become lost or confused during a complex sequence of operations.

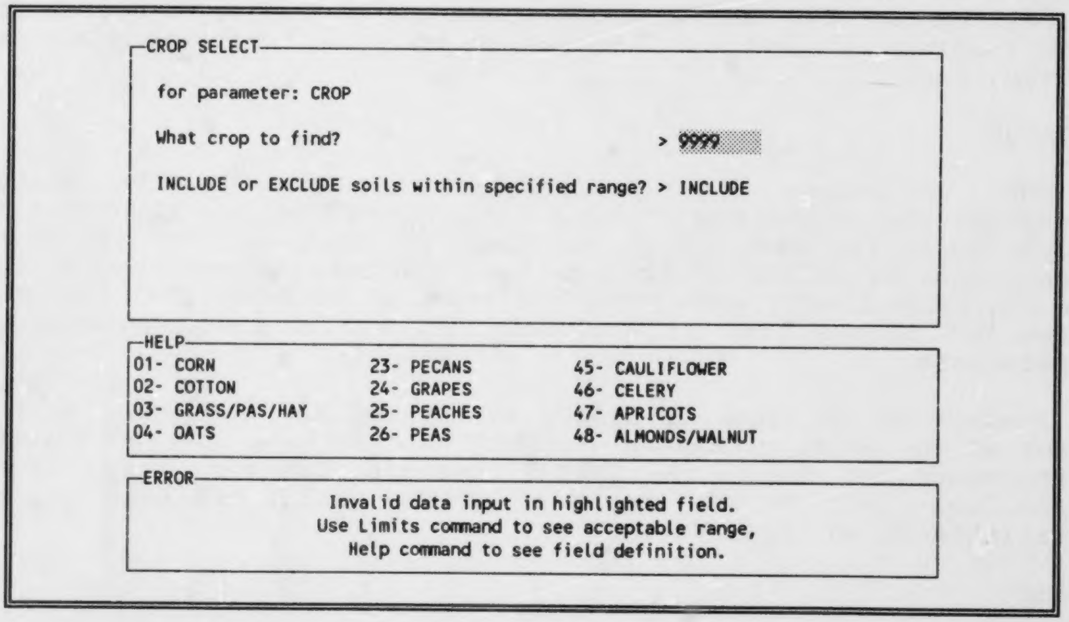
A maximum of 10 lines of STATUS assistance may be viewed by the user at any point within an application. Because the application programmer may change the STATUS messages, however, there is no practical limit to the amount of STATUS assistance that may be available in an application.

XPAD

Scratch pad (XPAD) assistance allows the user to write notes and reminders during an interactive session. The current capabilities in ANNIE-IDE for providing this type of assistance are of a basic nature. At this time the user may record information in a single XPAD with a maximum width of 80 characters and length of 10 lines. Regardless of where the user is located within the interactive session, a request for XPAD assistance will call up the same XPAD with the same information. ANNIE-IDE enables scrolling of XPAD information in the assistance window. New notes can be added to existing notes, and existing notes can be overwritten.

3.6 INSTRUCTION WINDOW

The **instruction window** is always present on every screen produced using ANNIE-IDE. In the screen layout, it is located below the data and assistance windows and directly above the command line (see Figure 3.1). The contents of this window are entirely controlled by the ANNIE-IDE software; the application programmer cannot modify the window contents. Two types of information are provided in the window: instructions for the user's next keystroke or error messages reporting incorrect keystrokes with instructions for corrective actions. Depending on which type of information is displayed by the system, the window title on the screen will be either "INSTRUCT" or "ERROR." Figure 2.3 gives an example of the type of information commonly provided in an INSTRUCT-type instruction window, and Figure 3.2 illustrates an ERROR-type instruction window.



```
CROP SELECT

 for parameter: CROP

 What crop to find?                               > 9999

 INCLUDE or EXCLUDE soils within specified range? > INCLUDE

HELP
01- CORN              23- PECANS        45- CAULIFLOWER
02- COTTON            24- GRAPES        46- CELERY
03- GRASS/PAS/HAY     25- PEACHES       47- APRICOTS
04- OATS              26- PEAS          48- ALMONDS/WALNUT

ERROR
              Invalid data input in highlighted field.
             Use Limits command to see acceptable range,
                Help command to see field definition.
```

Figure 3.2. Example error message in instruct window.

3.7 COMMAND LINE

The final component of the standard screen produced by ANNIE-IDE is the **command line** (Figure 3.1). The command line is restricted to one line having a maximum length of 80 characters. It contains a menu of abbreviations for the available commands at the user's current location within the program structure. The commands are automatically ordered by expected frequency of use, with the most commonly used commands located toward the beginning of the line. Definitions of the abbreviated commands are made available to the user by invoking the CMNDS assistance in the assist window (Section 3.6).

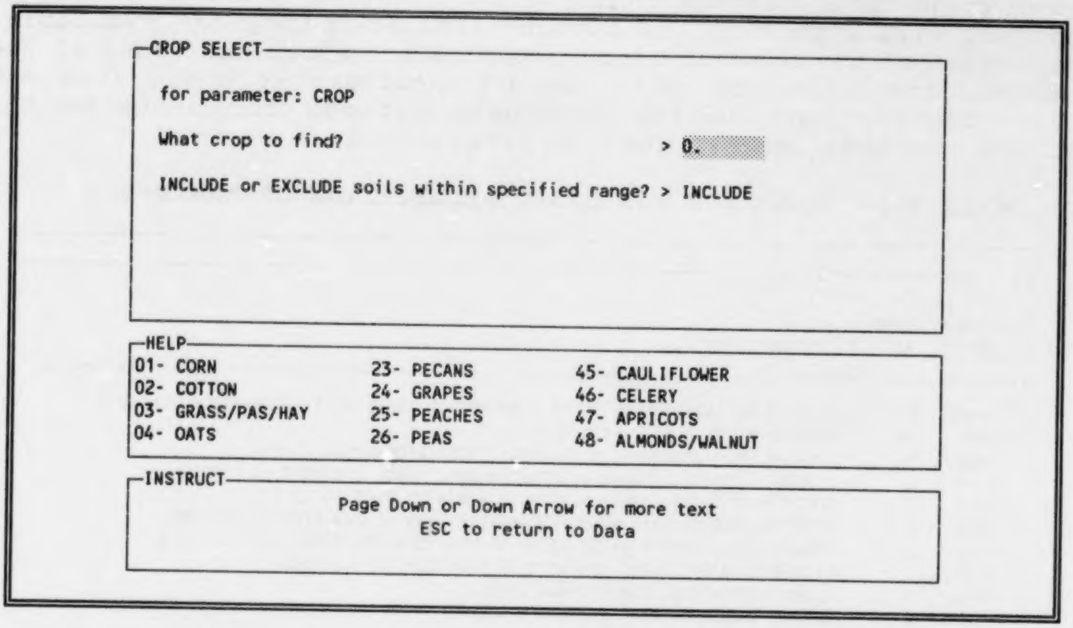
Table 3.3 lists the basic commands available for ANNIE-IDE applications. Included are definitions and command order numbers for eleven basic commands and three planned command extensions. The command order numbers indicate the relative order in which the commands will appear on the command line when they are available. Inspection of the command line in Figure 3.1 shows that some of the commands are associated with the PC functions keys and some are not. Instructions on the alternate methods for invoking the various commands are provided in Section 3.8.

TABLE 3.3. COMMANDS FOR BASIC APPLICATION OF ANNIE-IDE

COMMAND NAME	COMMAND ORDER #	COMMAND DEFINITION
CMNDS	2	DISPLAY DEFINITIONS OF VALID COMMANDS IN ASSISTANCE INFORMATION WINDOW
DNPG	14	DISPLAY NEXT PAGE IN DATA WINDOW
HELP	1	DISPLAY HELP INFORMATION IN ASSISTANCE INFORMATION WINDOW
LIMITS	6	DISPLAY LIMITS OF CURRENT FIELD IN ASSISTANCE INFORMATION WINDOW
NEXT	3	GO TO NEXT SCREEN (SETS SCREEN EXIT STATUS CODE TO 1)
OOPS	8	RESET VALUES IN DATA WINDOW TO VALUES WHEN SCREEN FIRST DISPLAYED
QUIET	20	TURN OFF ASSISTANCE INFORMATION WINDOW TO ALLOW MORE ROOM FOR DATA
STATUS	7	DISPLAY SYSTEM STATUS IN ASSISTANCE INFORMATION WINDOW
UPPG	15	DISPLAY PREVIOUS PAGE IN DATA WINDOW
WINDOW	9	DEFINE CORNER OF DATA OPERATION WINDOW
XPAD	21	DISPLAY USERS SCRATCH PAD, ALLOW CHANGES
planned command extensions		
MOVE	10	COPY ROWS/COLUMNS OF TWO-DIMENSIONAL DATA
TUTOR	5	PROVIDE TUTORIAL
VIEW	11	DISPLAY GRAPHIC DATA

It should be noted that the eleven basic commands are "fixed" within ANNIE-IDE, and cannot be removed by the programmer. As explained in the previous section not all eleven commands are available at any given location within the program, but their availability is determined by the system, not the application programmer. The programmer may, however, enable a number of additional commands related to movement between screens; these commands are introduced in Section 3.10.

A final feature of the command line is mentioned here to avoid confusion. As will be explained in the following section, three interaction modes are utilized by programs developed using ANNIE-IDE: data mode, command mode, and assist mode. The command line appears on the screen when the user is utilizing either the data mode or the command mode. When the user has invoked the assist mode, the command line is removed from the screen to avoid confusion, and command instructions are displayed in the instruction window. Figure 3.3 illustrates the layout for a screen when the user is in "assist mode" and the command line is not displayed. When the user leaves the assist mode to return to either of the other two modes, the command line reappears.



```
CROP SELECT

 for parameter: CROP

 What crop to find?                            > 0.

 INCLUDE or EXCLUDE soils within specified range? > INCLUDE

HELP
01- CORN              23- PECANS         45- CAULIFLOWER
02- COTTON            24- GRAPES         46- CELERY
03- GRASS/PAS/HAY     25- PEACHES        47- APRICOTS
04- OATS              26- PEAS           48- ALMONDS/WALNUT

INSTRUCT
                 Page Down or Down Arrow for more text
                        ESC to return to Data
```

Figure 3.3. Screen layout in assist mode.

3.8 USER CONTROL WITHIN SCREENS

3.8.1 User Interaction Modes

User interaction is organized into three "modes," each with a specific function:

- data mode - enter data or select from menu options in data window
- command mode - invoke commands or functions listed in command line; basic commands perform three functions:
  (1) allow exit from screens (NEXT)
  (2) manage assistance window (HELP, LIMITS, XPAD, STATUS, CMNDS, TUTOR, QUIET)

    - (3) manipulate data window (UPPG, DNPG, OOPS, WINDOW, MOVE, VIEW)
- assist mode - provide user with supplemental information in the assistance information window on which to base subsequent actions.

3.8.2 Keystrokes and Communication Features

ANNIE-IDE applications allow the user the flexibility of using both keyboard and mouse communication. From the keyboard mode, commands may be invoked either by pressing designated function keys or by typing the first letter of a command name. Likewise, menu options may be selected either by moving the cursor (either by use of cursor keys or a mouse) to the selection field and confirming, or by typing the first letter (or letters, if needed) of the menu item. For each of the three interaction modes, individual keys and key groups are used in a consistent manner as summarized in Table 3.4.

Several features of user communication using these keystrokes within the framework of ANNIE-IDE should be noted:

- there are no restrictions to upper- or lower-case mode
- the system does not use the same key or command to invoke two different functions
- cursor keys are always used to invoke movement within a screen
- cursor or mouse movement in command mode results in the sequential highlighting of command options; pressing the ENTER key, or clicking the mouse, invokes the currently highlighted command
- character keys can be used to select commands when in command mode
- character and number keys are used to enter data in a field or select a menu item when in data mode
- function keys are only used to invoke commands
- the TAB and F3 keys are left undefined to avoid system-related conflicts
- in data mode, arithmetic operands (+,-,*,/,**) can be used to modify values in 1- or 2-D arrays. For 1-D screens, cursor movement highlights fields for individual data values. An arithmetic operation can implemented by typing it into the appropriate field and then pressing the "ENTER" or "RETURN" key. For 2-D screens, the "WINDOW" command allows the user to perform global manipulations by establishing a highlighted field, and then typing the operation into this field. All values within the specified field are modified accordingly.

TABLE 3.4. KEYSTROKES AND THEIR RESULTS

MODE	KEY GROUP	KEY CODE	RESULT
DATA	PRINTABLE CHARACTER	<ANY>	TEXT SCREEN - NOTHING MENU SCREEN - SELECT OPTION(S) ASSOCIATED WITH CHARACTER NO CONFIRMATION NEEDED DATA SCREEN - CHARACTER INSERTED AT CURSOR POSITION
	UNPRINTABLE	ENTER OR RETURN	TEXT SCREEN - NOTHING MENU SCREEN - NOTHING DATA SCREEN - END FIELD, MOVE TO NEXT FIELD
		ESC	SWITCH TO COMMAND MODE
		<OTHER>	NOTHING
	CURSOR MOVEMENT	<ANY>	MOVE WITHIN DATA WINDOW AS APPROPRIATE
	FUNCTION	F3	ESC
		<OTHER>	EXECUTE ASSOCIATED COMMAND (F1-'Help', etc.)
COMMAND	PRINTABLE CHARACTER	<ANY>	EXECUTE COMMAND WITH CORRESPONDING FIRST LETTER IF AVAILABLE
	UNPRINTABLE CHARACTER	ENTER OR RETURN	SELECT HIGHLIGHTED COMMAND
		ESC	SWITCH TO DATA MODE
		<OTHER>	NOTHING
	CURSOR MOVEMENT	LEFT	HIGHLIGHT COMMAND TO THE LEFT
		RIGHT	HIGHLIGHT COMMAND TO THE RIGHT
		<OTHER>	NOTHING
	FUNCTION	F3	ESC
		<OTHER>	NOTHING
ASSIST	PRINTABLE CHARACTER	<ANY>	XPAD - CHARACTER INSERTED AT CURSOR POSITION OTHER - NOTHING
	UNPRINTABLE CHARACTER	ESC	SWITCH TO DATA MODE
		ENTER OR RETURN	XPAD - MOVE TO START OF NEXT LINE OTHER - NOTHING
		<OTHER>	NOTHING
	CURSOR	<ANY>	MOVE WITHIN ASSISTANCE WINDOW AS POSSIBLE

3.8.3 Movement Between Interaction Modes

User movement from each of the interaction modes to the other modes can be summarized as follows.

data mode to command mode - press ESC key
data mode to assist mode - press function key associated with appropriate type of assistance or go through command mode

command mode to data mode - press ESC key
command mode to assist mode - select appropriate type of assistance from options in command line

assist mode to data mode - press ESC key
assist mode to command mode - press ESC key twice (goes through data mode)

3.9 SCREEN DEFINITION DIRECTIVES

The content of screens is defined by using **screen definition directives**. The concept of **directives** is used in ANNIE-IDE to (1) eliminate the code required to construct screens from the main body of the interactive program and (2) enable the application programmer to use consistent methods for screen construction. The directives provide necessary screen definition information to the toolkit subroutines that control text and data display on the screens.

ANNIE-IDE allows the construction of five different screen types. The **screen types** and their functions are:

TEXT	display text
MENU	select an option from a menu of options
1-D ARRAY	edit a one-dimensional (1-D) array of related values
2-D ARRAY	edit a two-dimensional (2-D) array of related values
FILE	open a file to store or retrieve text and other information

Each of these screen types has its own list of directives and subdirectives (i.e., directives that are subordinate to the primary directives). The functions that are determined by the directives for each screen type are summarized in Figure 3.4. The instructions for their use are presented in Section 4.2 and their application is illustrated in Section 5.

The collection of directives and subdirective designations for a particular screen is called a **screen group**. Screen groups for all screens needed by the interactive program are constructed in a sequential file called a screen definition file by using a text editor. After this file has been developed, ANNIE-IDE provides a stand-alone interactive program called MESSIE, which is used to

TEXT SCREEN
(1) provide text for display
(2) define help-type assistance
(3) define screen name

MENU SCREEN
(1) provide menu title
(2) assign default menu option selection
(3) define length of menu options
(4) specify whether menu options are to be numbered
(5) define screen column width and length
(6) provide identification string for menu items
(7) provide description text for options
(8) define help-type assistance for each option
(9) define general help-type assistance for menu
(10) define screen name

FILE SCREEN

either
(1) define prompt asking for name of file
and/or
(2) define name of file (no number 1) or provide name for file name input field

(3) define file status
(4) specify file access method, format, and length of record
(5) define help-type assistance for screen
(6) define screen name

1-D and 2-D ARRAY SCREENS
(1) provide text explaining screen contents
(2) provide names for input parameter fields
(3) specify data types (e.g., INTEGER or REAL)
(4) assign default values for parameters
(5) define valid range for numeric parameters
(6) define valid parameter values for any type
(7) define invalid parameter values for any type
(8) specify storage instructions for screen group
(9) define help-type assistance for parameters
(10) define name of storage file for parameters
(11) provide text header and trailer for storage file
(12) define prompt for screen fill-in
(13) define help-type assistance for entire screen
(14) define screen name

ADDITIONAL 2-D ARRAY SCREEN FEATURES

(15) specify sorting requirements
(16) specify field protection

Figure 3.4. Screen directive functions.

convert the sequential file to a **WDM-message file**. The WDM-message file is direct access and hence enables more efficient retrieval of screen information. Further instructions on building and storing screen definition files are provided in Section 4.2.

3.10 SCREEN SEQUENCING

One of the most important functions of a good interactive program is providing the user with the ability to move from one screen to others in an expedient manner. The necessary types of interscreen movement vary from one program to another and hence need to be defined by the application programmer. The capabilities for interscreen commands have not been fully developed in ANNIE-IDE at this time. A list of useful interscreen commands has been developed, however, and a methodology for their development by the application programmer has been devised and will be explained as an "advanced" application of ANNIE-IDE in Section 4.4

Table 3.5 presents our list of optional commands for interscreen movement. The commands are defined, and the relative order in which ANNIE-IDE will sequence the commands on the screen command line (with respect to each other and the basic commands in Table 3.2) is indicated. The significance of the "screen exit status code" will be explained in Section 4.4.

TABLE 3.5. OPTIONAL COMMANDS FOR INTERSCREEN MOVEMENT

COMMAND NAME	COMMAND ID #	SCREEN EXIT STATUS CODE	COMMAND DEFINITION
ABORT	18	6	STOP EVERYTHING QUICKLY
BEGIN	12	3	GO TO FIRST SCREEN
EXIT	17	5	CLEAN UP AND RETURN TO OPERATING SYSTEM
GOTO	13	4	GO TO SCREEN TO BE SPECIFIED
INTRPT	16	7	STOP CURRENT ACTIVITY
NEXT	3	1	GO TO NEXT SCREEN (ALWAYS AVAILABLE)
PREV	4	2	GO TO PREVIOUS SCREEN

SECTION 4

INSTRUCTIONS FOR USING ANNIE-IDE

The emphasis of the material presented in the previous section centered on the scope and features of ANNIE-IDE. In Section 4 the focus shifts to the mechanics of using the ANNIE-IDE software to build interactive applications. The goal of this section is to provide the procedural "how-to"s needed to implement the software's capabilities. To a large extent, the topics contained in the latter portions of Section 3 are reintroduced and discussed with much greater instructional detail.

The first two subsections present instructions for producing the two major components of an ANNIE-IDE application as depicted in Figure 4.1: the **MESSAGE.WDM file** and the **application program.** Section 4.1 provides instructions for producing and storing the user interaction text and allowable field value ranges by using screen directives, groups, and clusters; screen definition files; and WDM-message files. In Section 4.2, general guidance is given for implementing the application program. Further details are provided on the toolkit subroutines that are available to implement screen interaction control. Finally, Section 4.3 contains guidelines for using ANNIE-IDE to develop more complex application programs by implementing advanced capabilities related to interscreen movement and system status reporting. Throughout Section 4, frequent references are made to Section 5 which provides a full range of code, directives, and screen examples to support the development instructions.

4.1 BUILDING AND STORING SCREEN DEFINITION FILES

The content of each ANNIE-IDE screen is defined and processed as shown in Figure 4.2. The application programmer starts with ideas about the number and content of each screen required by the application. The content of each screen should be sketched out on paper. Information from the sketches is then translated into a series of **screen definition directives.** Screen definition directives consist of a keyword and an associated value. The associated value may be a text string, a number, or a series of numbers.

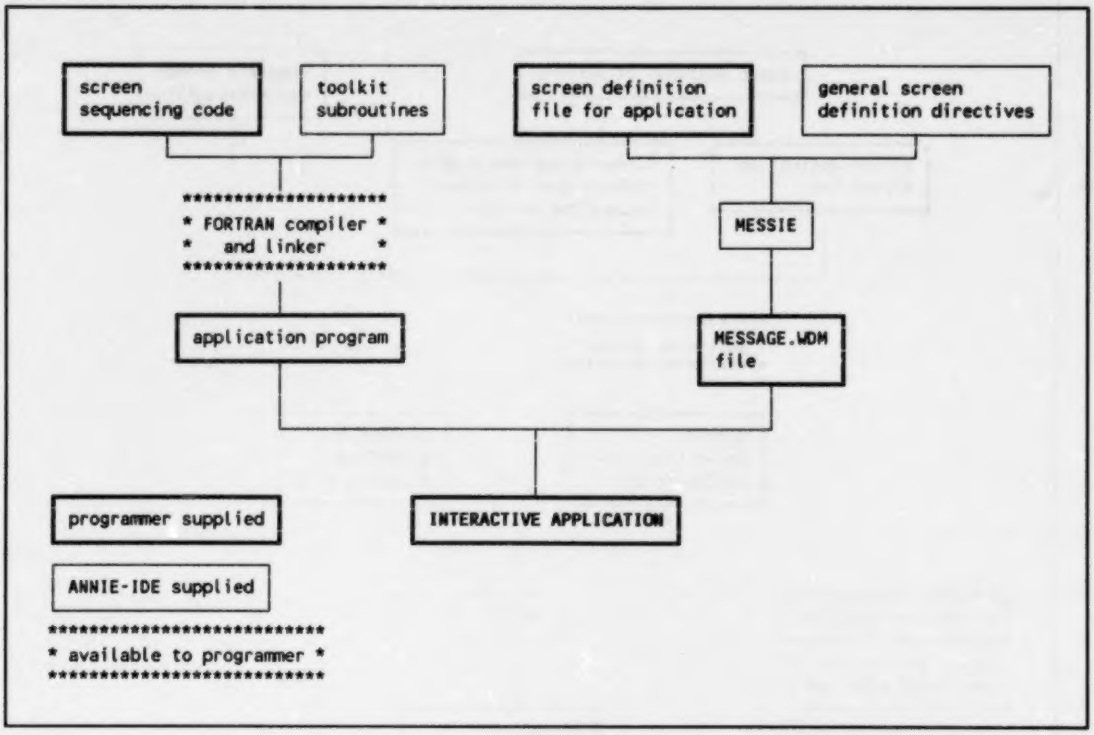

screen sequencing code
toolkit subroutines
screen definition file for application
general screen definition directives
* FORTRAN compiler and linker *
MESSIE
application program
MESSAGE.WDM file
INTERACTIVE APPLICATION
programmer supplied
ANNIE-IDE supplied
* available to programmer *

Figure 4.1. Building an interactive application.

The series of directives that define a single screen is called a **screen group**. A set of related screen groups is called a **screen cluster**. Related screen groups are aggregated into screen clusters in order to make efficient use of the long length of WDM-message file records. Each group and cluster has a unique identification number (ID) associated with it. These IDs are used within the application program to identify a particular screen. A screen cluster corresponds to a WDM dataset. One or more screen clusters are stored together in a **screen definition file**.

The screen definition file is a formatted, ASCII, sequential file in ANNIE-WDM import/export format containing directives that describe all the particulars of one or more clusters. The file may be created or edited using any text editor that does not add special characters or reformat the text. (If the editor works with source code, it will work with this file.)

A template screen definition file is shown in Figure 4.3. The template file is found in file 'SDEF.TEM' on the distribution

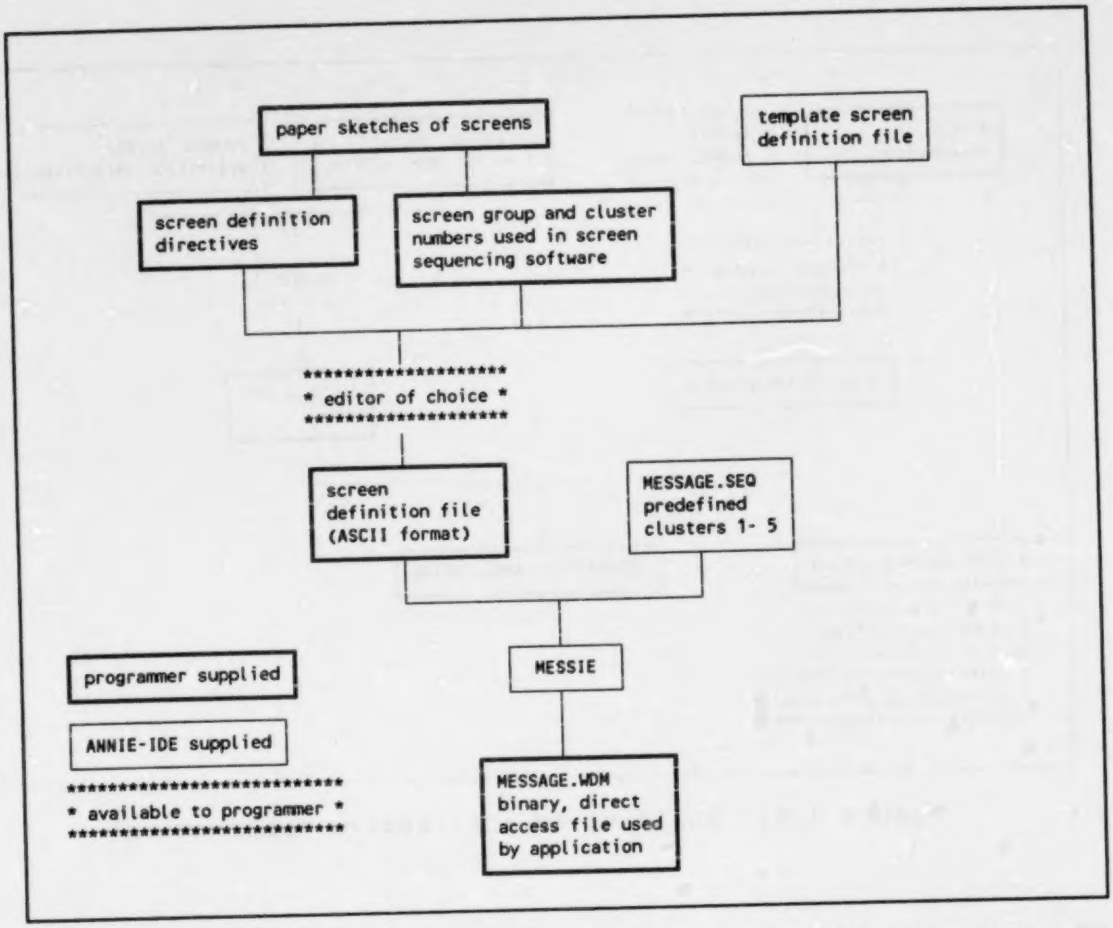

paper sketches of screens
template screen definition file
screen definition directives
screen group and cluster numbers used in screen sequencing software
****
* editor of choice *
****
screen definition file (ASCII format)
MESSAGE.SEQ predefined clusters 1- 5
MESSIE
programmer supplied
ANNIE-IDE supplied
****
* available to programmer *
****
MESSAGE.WDM binary, direct access file used by application

Figure 4.2. Building and storing a screen definition file.

disks. Complete screen definition files for DBAPE and ANNIE-QUAL2 are found on the ANNIE-IDE distribution disks.

Note that four-character abbreviations are used to identify the type of a screen group (fields 'd,' 'f' and 'j' in Figure 4.3:

TEXT - text
MENU - menu
PRM1 - 1-dimensional parameter
PRM2 - 2-dimensional parameter
FILE - file specifications

All screens included in a particular application are stored in a WDM-message file. One or more screen definition files are converted into an application's WDM-message file using the stand alone program "MESSIE." The WDM-message file is an unformatted

```
DATE
WDMSFL
SYSTEM
COMMENT
  s
END COMMENT
CLU         a      TYPE  MESS   NDN   1   NUP   1   NSA   5   NSP  20   NDP 400
  LABEL
    GRPNAM   b
  END LABEL
  DATA   CLU   a
#GROUP   c   TYPE  d

< screen definition directives for group c of cluster a >

#GROUP   e   TYPE  f

< screen definition directives for group e of cluster a >

  END DATA
END CLU
CLU         g      TYPE  MESS   NDN   1   NUP   1   NSA   5   NSP  20   NDP 400
  LABEL
    GRPNAM   h
  END LABEL
  DATA   CLU   g
#GROUP   i   TYPE  j

< screen definition directives for group i of cluster g >

  END DATA
END CLU


notes:

application programmer fills in highlighted fields of template as follows:

a  - cluster number of first cluster in file
b  - cluster name
c  - group number of first screen in cluster
d  - type of first screen
e  - group number of second screen in cluster
f  - type of second screen
g  - cluster number of second cluster in file
h  - cluster name
i  - group number of first screen in second cluster
j  - type of first screen in second cluster
s  - optional comment which describes this screen definition file
```

Figure 4.3. Template screen definition file.

direct-access file with a record length of 2048 bytes. WDM utility routines keep a buffer of records from the file in memory in order to reduce the number of read requests made to the file. This allows for quicker loading of screen information from the WDM-message file. The WDM-message file is always named "MESSAGE.WDM."

Cluster numbers one through five are reserved for internal use by ANNIE-IDE and are contained in a file named "MESSAGE.SEQ" provided on the distribution disk.

Screen directives are organized hierarchically - the "$" keyword prefix indicates a primary directive associated with an entire screen; the "_" keyword prefix indicates a subdirective associated with a field or option. The following sections describe the individual directives associated with each screen type.

4.1.1 Text Screen Directives

A TEXT screen is defined using the directives shown in Table 4.1. Note that "&" is used to define where optional numeric information is to be written within the text. The optional numberic information is provided as subroutine arguments. Example TEXT screens with associated directives are shown in Sections 5.4 and 5.5.

TABLE 4.1. TEXT SCREEN DEFINITION DIRECTIVES

KEYWORD	POS	NOTES
$WINDOW	aft	name of screen (up to 48 characters)
$TEXT	nxt	text to display is data window of screen, total number of lines must be 50 or less, max length of line to display in data window is 78, use "&" to indicate spot where toolkit text screen display routine is to fill in a value
$HELP	nxt	general help for screen

NOTES:

POS indicates where to specify value for directive as follows:
1) nxt - text on following records
2) aft - text after keyword

4.1.2 Menu Screen Directives

Screen definition directives to define a MENU screen are shown in Table 4.2. An example MENU screen with accompanying directives is shown in Section 5.6.

TABLE 4.2. MENU SCREEN DEFINITION DIRECTIVES

KEYWORD	POS	NOTES
$WINDOW	aft	window name (up to 48 characters)
$TITLE	aft	title of menu, maximum length 78 characters, appears on first line of data window
$DEFAULT	afi	default option; if omitted, no default
$LENGTH	afi	length of longest option identification string; required, must be between 1 and 63
$WIDTH	afi	column width, default 78
$COLENGTH	afi	length of each column, must be between 1 and 8; default 8
$OPTION	aft	option identification string,length must be less than or equal to $LENGTH
_DESC	aft	description of option, length must be less than ($WIDTH - $LENGTH - 3)
_HELP	nxt	help for option, use as many records as required, up to 78 characters per record
$HELP	nxt	general help related to all options, as many records as required, up to 78 characters per record

NOTES:

POS indicates where to specify value for directive as follows:
1) nxt - text on following records
2) aft - text after keyword
3) afi - integer after keyword
4) flg - flag, no value needed

4.1.3 1-D Array (PRM1) Screen Directives

A one-dimensional array (PRM1) screen is defined using the directives shown in Table 4.3. Example PRM1 screens with accompanying directives are found in Sections 5.8 and 5.14.

TABLE 4.3. PRM1 SCREEN DEFINITION DIRECTIVES

KEYWORD	POS	NOTES
$WINDOW	aft	window name (up to 48 characters)
$SCREEN	nxt	up to 10 records of text which represent the screen menu. field names are preceded by '@' and may be extended by one or more '.' in order to make field wide enough for expected values, up to 30 fields may be indicated
$HELP	nxt	general help for screen
$FIELD	aft	name of field, must match name in $SCREEN (including '@', excluding '.')
_TYPE	aft	data type - INTEGER, REAL, DOUBLE PRECISION, or CHARACTER (type must be specified before any other subdirectives under $FIELD)
_PROTECT	aft	field protection - NONE (default), CORRECT (must be in valid range), PROTECTED (cannot change value)
_RANGE	aft	minimum and maximum value allowed for numeric fields, separated by a colon, -999 if unknown
_VALID	aft	valid values for any type of field, separated by commas
_DEFAULT	aft	default value for parameter (must appear after _RANGE and _TYPE)
_INVALID	aft	invalid values for any type of field
_HELP	nxt	help information about field
*warning, the following optional directives have not been tested within the 1-D screen routines		
_RECORD	afi	record number in output data file where value will be stored
_SLOT	afi	start column in output data file record where value will be stored
_PCODE	afi	parameter code for field
_UCODE	afi	units code for field
_FORMAT	aft	format for field output to file
$OUTPUT	aft	name of file to store field values on
_HEADER	aft	character string written at beginning of output file
_TRAILER	aft	character string written at end of output file

NOTES:

POS indicates where to specify value for directive as follows:
1) nxt - text on following records
2) aft - text after keyword
3) afi - integer after keyword

4.1.4 2-D Array (PRM2) Screen Directives

A two-dimensional array (PRM2) screen is defined using the directives shown in Table 4.4. An example PRM2 screen with accompanying directives is found in Section 5.9

TABLE 4.4. PRM2 SCREEN DEFINITION DIRECTIVES

KEYWORD	POS	NOTES
$WINDOW	aft	window name (up to 48 characters)
$HEADER	nxt	header for screen (may be multiple lines)
$HELP	nxt	general help for screen
$FIELD	aft	name of input array field (no '@' required as in 1-D)
*next 5 directives required before other remaining directives		
_TYPE	aft	data type - INTEGER, REAL, DOUBLE PRECISION, or CHARACTER
_WIDTH	afi	field width in characters
_ORDER	aft	sort requirement - RANDOM (default), ASCENDING, DESCENDING
_PROTECT	aft	field protection - NONE (default), CORRECT (must be in valid range), PROTECTED (cannot change value)
_COLUMN	afi	starting column of field
_HELP	nxt	help information for field
_RANGE	aft	valid range for numeric field
_VALID	aft	valid values for any type field
_DEFAULT	aft	default value for field
_INVALID	aft	invalid values for any type field
*warning, the following optional keywords have not been tested within the 2-D screen routines		
_PACK	afi	number of values to pack on a data record
_FORMAT	afi	format of values on data record
_PCODE	afi	parameter code for field
_UCODE	afi	units code for field
_FORMAT	aft	format for field output to file
$OUTPUT	aft	name of file to store field values on
_HEADER	aft	character string written at beginning of output file
_TRAILER	aft	character string written at end of output file

NOTES:

POS indicates where to specify value for directive as follows:
1) nxt - text on following records
2) aft - text after keyword
3) afi - integer after keyword

4.1.5 File Screen Directives

A FILE screen is defined using the directives shown in Table 4.5. An example FILE screen with accompanying directives is found in Section 5.7.

TABLE 4.5. FILE SCREEN DEFINITION DIRECTIVES

KEYWORD	POS	NOTES
$WINDOW	aft	window name (up to 48 characters)
$SCREEN	aft	screen menu text when asking for name of file, up to 10 records, field name preceded by '@' and may be extended by one or more '.', this keyword used only if file name is to be provided by user
$NAME	aft	name of file ($SCREEN keyword not allowed) provided by application or field name in screen text directive which will be provided by user
$STATUS	aft	file status, OLD (default), NEW, UNKNOWN, or SCRATCH
$ACCESS	aft	file access method, SEQUENTIAL (default) or DIRECT
$FORM	aft	file format, FORMATTED (default) or UNFORMATTED
$RECL	aft	length of record in direct access file (bytes if formatted, half words if unformatted)
$HELP	nxt	help information for screen

NOTES:

POS indicates where to specify value for directive as follows:
1) nxt - text on following records
2) aft - text after keyword
3) afi - integer after keyword

4.1.6 Using MESSIE

MESSIE is a stand-alone interactive program that is used by the application programmer to import and export screen definition data between the screen definition file format used during development and the direct-access format used during interactive program execution. MESSIE's role comes into play after the application programmer has constructed screen groups and clusters for the screens needed by the interactive program. The screen groups are entered in a screen definition file by using a text editor. When the programmer is ready to test the interactive program, it is necessary to use MESSIE to convert the screen definition file to a WDM-message file.

It will most likely be necessary during the development effort for the application programmer to modify existing screens and add new ones. To accomplish these tasks efficiently, MESSIE also can be used to export the WDM-message file contents back to a screen definition file, which can be modified by a text editor, and then re-imported to MESSAGE.WDM.

Instructions for using MESSIE to create the WDM-message file, import screen definition data to it from a sequential file, and export screen groups out of the message file back to a sequential file are provided below.

Create WDM-message File

A prerequisite to using MESSIE to create a WDM-message file is that the file named "MESSAGE.SEQ" be copied to the directory in which MESSAGE.WDM is to be created. The MESSAGE.SEQ file is provided as part of the distributed software for ANNIE-IDE. This screen definition file contains standard messages and information needed for all applications of ANNIE-IDE. When the file is available to MESSIE, the following steps should be taken (on a PC):

(1) Type "MESSIE" to execute the MESSIE.EXE file. MESSIE will try to open the MESSAGE.WDM file. Unable to find it, MESSIE automatically opens a new file by that name and reports a successful open. Next, MESSIE automatically imports the MESSAGE.SEQ file into MESSAGE.WDM. The message file is now ready for import of screen clusters constructed by the application programmer.

(2) If no programmer-generated screen clusters are to be imported at this time, instruct MESSIE to return to the operating system by typing "R."

Import a Screen Definition File

Prerequisite: Successful creation of WDM-message file and import of MESSAGE.SEQ file

Instructions:

(1) Execute program MESSIE. On a PC type "MESSIE" to execute the MESSIE.EXE file. Program will report successful open of MESSAGE.WDM file or an attempt will be made to create the MESSAGE.WDM file as described above.

(2) Following prompt, select import option by entering "I".

(3) Following prompt, enter file name for screen definition file. MESSIE will automatically begin to import the individual screen clusters contained in the specified screen definition file; the successful import of each screen cluster will be reported on the screen.

If MESSIE encounters screen clusters in the screen definition file that are already contained in MESSAGE.WDM (as identified by comparing screen cluster numbers), the program asks whether to skip the screen cluster (i.e., don't import it), overwrite the screen cluster already in the WDM-message file, or abort the entire import process. Appropriate responses are "S," "O," and "A," respectively.

When MESSIE has completed the import of screen clusters in the screen definition file, the end of the importing process is reported on the screen.

(4) If no other import tasks are needed, enter "R" following prompt to return to the operating system.

Export WDM-message File to Sequential File

Prerequisite: WDM-message file must contain something to export (i.e., screen clusters)

Instructions:

(1) Execute "MESSIE." Program will report successful open of MESSAGE.WDM.

(2) When prompted, select export option by entering "E".

(3) When prompted, enter a name for the output file. The name must not already exist in the current directory.

(4) MESSIE prompts for comment lines that will be used to label the output file. Type in comment lines - use "ENTER" with no text to end the comment lines.

(5) MESSIE asks whether all or selected screen groups are to be exported from the MESSAGE.WDM file. If all screen clusters are to be exported, enter "A." MESSIE will automatically export all clusters contained in the WDM-message file to the output file.

If only selected screen clusters are to be exported, enter the screen cluster identification numbers for these clusters one-at-a-time as they are prompted. Successful completion of export for each screen cluster will be reported by MESSIE.

(6) When exports have been completed, enter "0" (zero). Instruct program to return to the operating system by typing "R."

4.1.7 Directives to Provide Assistance to User

The concept of layered assistance to the user was introduced in Section 3.5. Screen directives used to supply user assistance at the three assistance levels for each screen type are shown in Table 4.6.

TABLE 4.6. SCREEN DIRECTIVES FOR SPECIFYING ASSISTANCE

	SCREEN TYPE				
ASSISTANCE LEVEL	TEXT	MENU	PRM1	PRM2	FILE
1 - field or option name	none	$OPTION	$SCREEN	$HEADER	none
2 - additional information in DATA window	none	_DESC	$SCREEN	$HEADER	$TEXT
3 - additional information in ASSISTANCE window	$HELP	_HELP	_HELP _VALID _INVALID _RANGE $HELP	_HELP _VALID _INVALID _RANGE $HELP	$HELP _VALID

4.2 DESIGNING AND DEVELOPING THE INTERACTIVE PROGRAM

The ANNIE-IDE package can be utilized to build a user interface for virtually any type of computer program or computer-stored data base. Potential applications range from generating user-supplied input for already existing environmental or scientific models to allowing user access to a variety of data bases.

The structure of an application program designed to provide a user interface for a computer program is shown in Figure 4.4.

Double-lined box borders are used to show functions that are the responsibility of the application programmer. Single-lined box borders are used to show ANNIE-IDE toolkit functions. Sections 4.2.1 through 4.2.7 describe the application program functions introduced in Figure 4.4. Example Fortran codes to perform the functions are found in Section 5.

A major portion of the programmer's effort in developing an interactive program will be devoted to specifying the details of screen interaction control. Section 4.2.8 gives further instruction on using toolkit routines for this purpose. Closing comments on the interactive program development process are provided in Section 4.2.9.

4.2.1 Initialize ANNIE-IDE

Before any screens are displayed the application program must initialize ANNIE-IDE. Initialization tasks include opening the WDM-message file named "MESSAGE.WDM," reading default system parameter information from the WDM-message file, determining user-specified system parameters by reading the TERM.DAT file, and reading the old scratch pad file XPAD.DAT into the applications scratch pad. Example code demonstrating this function is shown in section 5.1.

4.2.2 Initialize Parameters

Most environmental models have data structures that contain parameters required to execute the model. The purpose of this function is to initialize these parameter values. If no default value is appropriate, then the value should be set to an arbitrary value indicating that it is undefined. In ANNIE-QUAL2 the undefined value is -999. This function is required because the user may choose not to enter data for each of the data entry screens defined for the model. When that situation occurs, the parameters for the model must either be reasonable or cause a fatal error. This function is demonstrated with Fortran code found in Section 5.2.

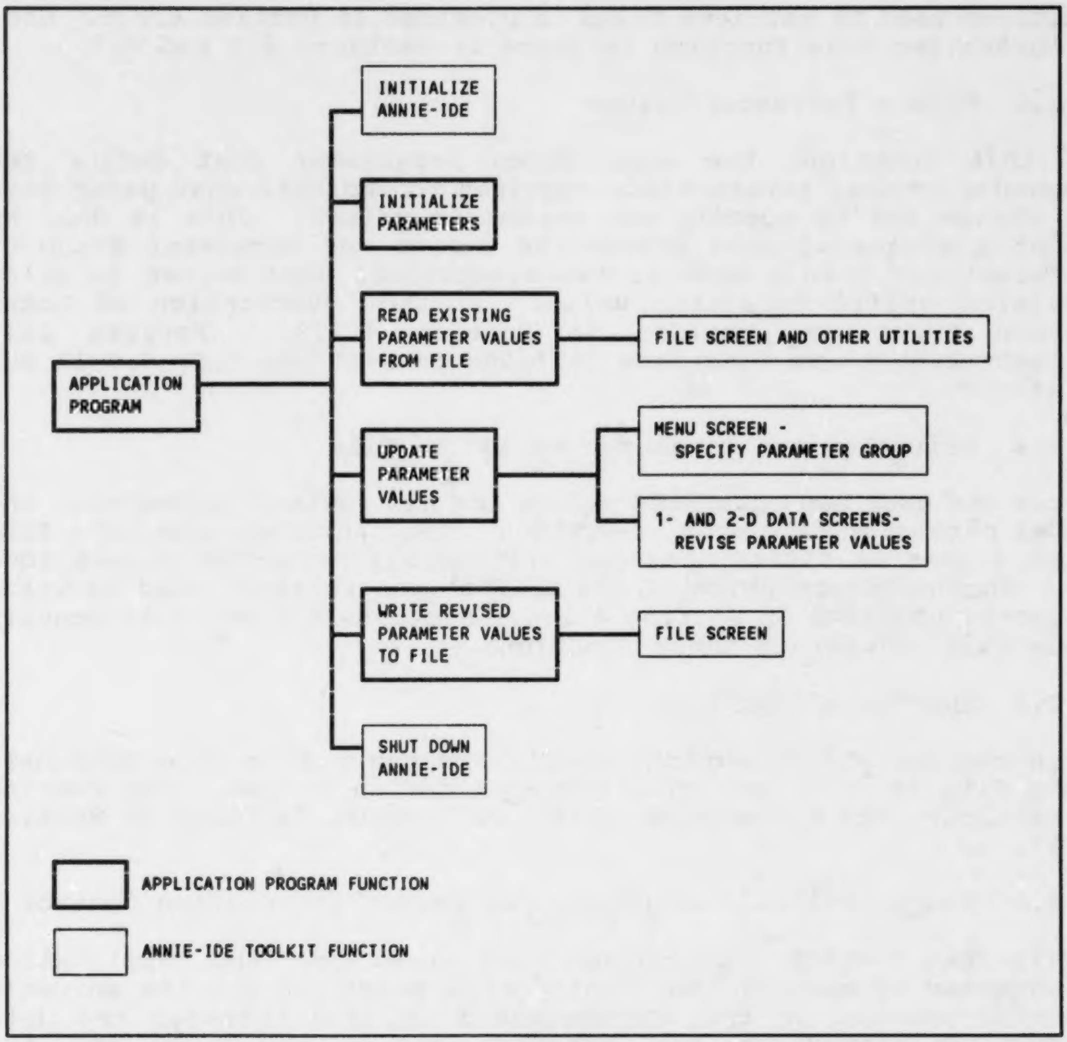

INITIALIZE ANNIE-IDE
INITIALIZE PARAMETERS
READ EXISTING PARAMETER VALUES FROM FILE
FILE SCREEN AND OTHER UTILITIES
APPLICATION PROGRAM
UPDATE PARAMETER VALUES
MENU SCREEN - SPECIFY PARAMETER GROUP
1- AND 2-D DATA SCREENS- REVISE PARAMETER VALUES
WRITE REVISED PARAMETER VALUES TO FILE
FILE SCREEN
SHUT DOWN ANNIE-IDE
APPLICATION PROGRAM FUNCTION
ANNIE-IDE TOOLKIT FUNCTION

Figure 4.4. Structure of interactive program providing user interface to an environmental model.

4.2.3 Read Existing Parameter Values From a File

The user may be executing a series of model runs as part of model calibration or an examination of alternatives. In this case, current model parameters must be retrieved from an existing file. The interactive program must not fail due to a read error caused by an error in the file. Further description of the FILE screen routines used to retrieve files is provided in Section 4.2.8. Code illustrating this function is found in Sections 5.3 and 5.7.

4.2.4 Update Parameter Values

In this function, the application programmer must define the sequence of user interactions required to indicate what parameters to change and to specify new parameter values. This is done by using a series of menu screens to define the parameter group of interest and then a one- or two-dimensional data screen to allow revision of the parameter value. Further description of these screen types is provided in Section 4.2.8. Fortran code illustrating these functions is found in Sections 5.6, 5.9-12 and 5.14.

4.2.5 Write Revised Parameter Values To File

After the user has specified values for all revised parameters, the model parameter file must be written. This involves opening a file with a user specified name and writing all parameter values into it. Further description of the FILE screen routines used to write files is provided in Section 4.2.8. Sections 5.7 and 5.16 contain code that illustrate these functions.

4.2.6 Shut Down ANNIE-IDE

This routine writes out the user's scratch pad to file XPAD.DAT. This file is read next time the application is run. The routine also closes the WDM-message file. An example is found in Section 5.17.

4.2.7 Using Toolkit Subroutines for Screen Interaction Control

ANNIE-IDE toolkit subroutines are used by the application programmer to control the display of screens, locate the screen's general content on the WDM-message file, and transfer the data values displayed on the screens between the user and the application program (and vice versa). A call from the application program to any of the toolkit screen interaction routines described in this section results in the display of a screen. The general content of the screen is defined by the specification of the screen group and cluster (Section 4.1) on the WDM-message file in the toolkit argument list. Data values displayed on the screen and edited by the user are transferred between the screen and the application through additional arguments in the toolkit routines.

The ANNIE-IDE toolkit subroutines are organized into five groups, each of which allows interaction with one of the five screen types defined in Section 3.9. Descriptions of the function of each group follow, showing the relationship between the toolkit routines with a structure chart, and providing general instructions for use of the routines. The structure charts represent the heirarchy of the various routines within a group. Their purpose is to enable the programmer to locate a routine within its program context so that its relationship to the rest of the group can be better understood. Each box within the structure chart represents a single subroutine. Subroutines call other subroutines that are directly below and connected by a line. The higher level routines perform the more general functions, and the lower level routines perform specific functions. Routines shown on the same level perform functions of similar detail.

All the subroutines shown in this section's structure charts are "lead" subroutines. These subroutines call additional subordinate subroutines contained within the ANNIE-IDE software. The subordinate subroutines accomplish specific tasks required for the performance of the more general functions of the lead routines. Default values for the arguments of the subordinate routines are provided by code in the lead routines and hence they are able to perform their tasks for basic applications in a transparent fashion. If the experienced application programmer wishes to "fine tune" the screens produced by a group, however, it may become necessary to understand the optional capabilities allowed by the subordinate routines. To this end, Appendix B.2 lists all available routines and their functions, and the file SYSDOC.OUT contained on the distribution disk provides details on their use. Example applications of routines from each group are found in Section 5. Details of the subroutine arguments are provided in Appendix B.1.

TEXT Display Subroutines

TEXT screens are used to display information to the user. A screen may contain up to 50 lines of text, with 10 or 16 lines available for viewing in the data window at any time. The user may use the cursor movement keys to view portions of text not displayed in the data window. The text may contain character or numeric data values supplied by the application program through the subroutine's argument list. Generally the text displayed on the screen is retrieved from the MESSAGE.WDM file. However, if the text varies significantly within a screen (as with model results), then the application programmer may specify the entire contents of the screen within the application program.

Figure 4.5 contains a structure chart showing the relationships among the various text display routines. Three general classes of text display routines are identified in the structure chart by the number that follows the routine name. Routines within each class are differentiated by the descriptions in Figure 4.6.

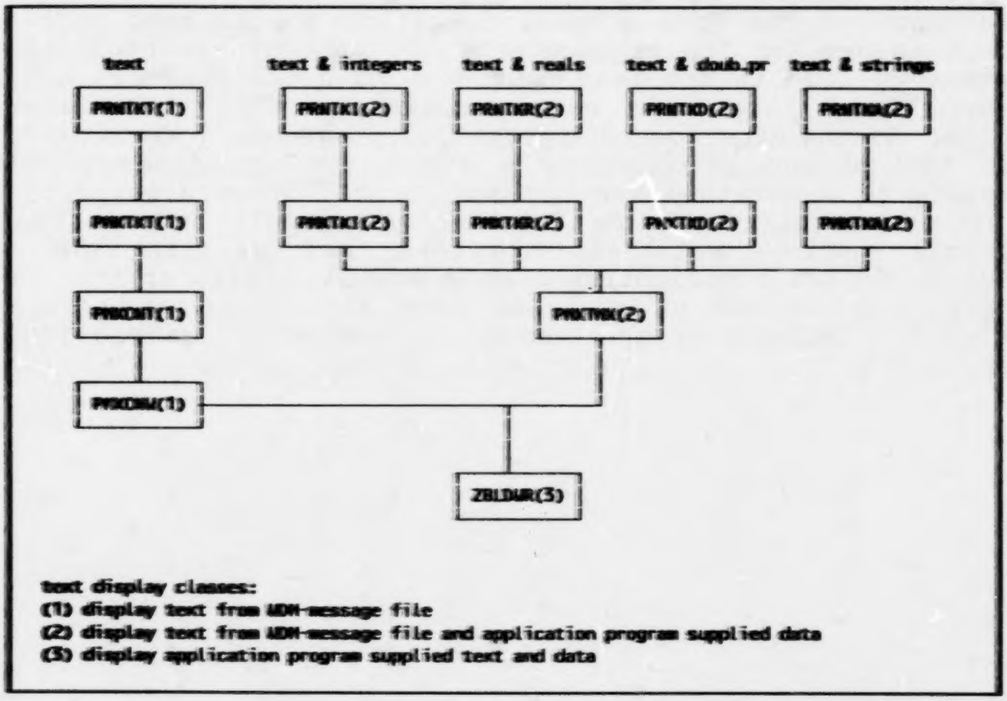

text
text & integers
text & reals
text & doub.pr
text & strings
PRNTXT(1)
PRNTXI(2)
PRNTXR(2)
PRNTXD(2)
PRNTXA(2)
PMXTXT(1)
PMXTXI(2)
PMXTXR(2)
PMXTXD(2)
PMXTXA(2)
PMXCNT(1)
PMXTXM(2)
PMXCNW(1)
ZBLDWR(3)
text display classes:
(1) display text from WDM-message file
(2) display text from WDM-message file and application program supplied data
(3) display application program supplied text and data

Figure 4.5. Structure chart for text display subroutines.

SUBROUTINE	PRIMARY ACTIONS
(TEXT FROM WOW-MESSAGE FILE ONLY)	
PRNTXT	DISPLAY TEXT FROM WOW-MESSAGE FILE, WRITE AND WAIT, INITIALIZE SCREEN
PMXTXT	SAME AS PRNTXT, OPTIONALLY INITIALIZE SCREEN, ALSO COUNTS LINES WRITTEN FOR RETURN TO APPLICATION PROGRAM
PRNCNT	SAME AS PMXTXT, COUNTS LINES WRITTEN FOR RETURN TO APPLICATION PROGRAM
PRNCNW	SAME AS PRNCNT, OPTIONALLY WRITE AND GO, WRITE AND WAIT FOR USER TO RESPOND, DON'T WRITE
(TEXT FROM WOW MESSAGE FILE AND DATA FROM APPLICATION PROGRAM)	
PRNTXI	DISPLAY TEXT FROM WOW-MESSAGE FILE, INTEGER VALUE FROM APPLICATION PROGRAM, WRITE AND WAIT, INITIALIZE SCREEN
PMXTXI	SAME AS PRNTXI, MAY WRITE MANY INTEGERS
PRNTXR	SAME AS PRNTXI EXCEPT WRITES REAL
PMXTXR	SAME AS PRNTXR, MAY WRITE MANY REALS
PRNTXD	SAME AS PRNTXI EXCEPT WRITES DOUBLE PRECISION
PMXTXD	SAME AS PRNTXD, MAY WRITE MANY DOUBLE PRECISIONS
PRNTXA	SAME AS PRNTXI EXCEPT WRITES CHARACTER STRINGS
PMXTXA	SAME AS PRNTXA, MAY WRITE MANY CHARACTER STRINGS
PMXTXM	DISPLAY TEXT FROM WOW-MESSAGE FILE WITH A COMBINATION OF INTEGERS, REALS, DOUBLE PRECISIONS, AND STRINGS
(TEXT AND DATA FROM APPLICATION PROGRAM)	
ZBLDWR	DISPLAY PROGRAMMER SPECIFIED TEXT AND DATA, OPTIONALLY WRITE AND GO, WRITE AND WAIT FOR USER TO RESPOND, DON'T WRITE; RETURNS USER SCREEN EXIT COMMAND CODE

Figure 4.6. Text display subroutines.

Details of the arguments required in order for these routines to be called from an application program are found in Appendix B.1.

MENU Selection Subroutines

MENU screens are used to display a series of options and prompt the user to request one. No more than seven options should be displayed on a particular screen. The application programmer is allowed to display up to 64 options if needed, however.

Two toolkit routines are used for menu selection. They are shown in Figure 4.7. Both return the order number of the user's response (option one returns 1, etc.); QRESPS also returns the text of the selected option.

Details of the arguments required in order for these two routines to be called from an application program are found in Appendix B.1.

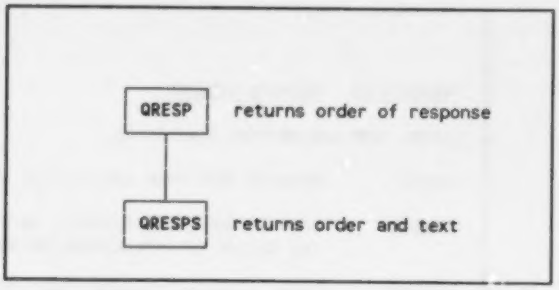

QRESP
returns order of response
QRESPS
returns order and text

Figure 4.7. Structure chart for menu subroutines.

ONE-DIMENSIONAL Array Edit Subroutines

The one-dimensional data edit subroutines are used to prompt the user to select options, enter numeric values, and/or enter text. Up to 30 field entries may be used. User input is checked against valid values and errors are reported for incorrect input. Figure 4.8 contains a structure chart showing the relationships among the various data editing subroutines.

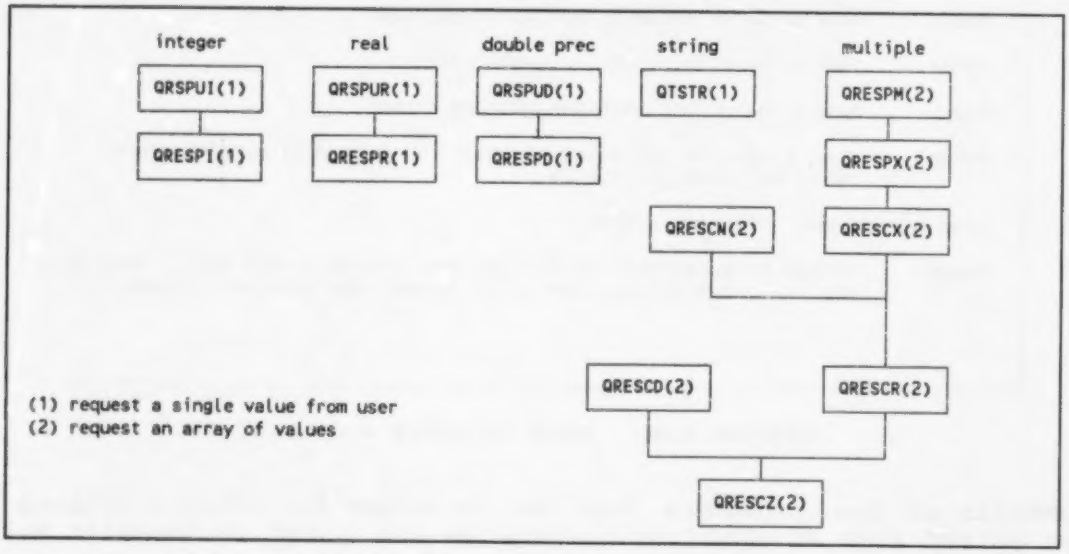

integer
real
double prec
string
multiple
QRSPUI(1)
QRSPUR(1)
QRSPUD(1)
QTSTR(1)
QRESPM(2)
QRESPI(1)
QRESPR(1)
QRESPD(1)
QRESPX(2)
QRESCN(2)
QRESCX(2)
QRESCD(2)
QRESCR(2)
QRESCZ(2)
(1) request a single value from user
(2) request an array of values

Figure 4.8. Structure chart for 1-D array subroutines.

There are two general classes of 1-D data edit routines identified on the structure chart by the number that follows the routine name. Routines within each class are differentiated by descriptions in Figure 4.9.

SUBROUTINE	PRIMARY ACTIONS
(SINGLE VALUE INPUT)	
QRSPUI	RETURN A SINGLE INTEGER VALUE, ALLOWABLE RANGES FROM APPLICAITON PROGRAM
QRESPI	RETURN A SINGLE INTEGER VALUE, ALLOWABLE RANGES FROM WDM-MESSAGE FILE
QRSPUR	RETURN A SINGLE REAL VALUE, ALLOWABLE RANGES FROM APPLICATION PROGRAM
QRESPR	RETURN A SINGLE REAL VALUE, ALLOWABLE RANGES FROM WDM-MESSAGE FILE
QRSPUD	RETURN A SINGLE DOUBLE PRECISION VALUE, ALLOWABLE RANGES FROM APPLICATION PROGRAM
QRESPD	RETURN A SINGLE DOUBLE PRECISION VALUE, ALLOWABLE RANGES FROM WDM-MESSAGE FILE
QTSTR	RETURN A TEXT STRING
(MULTIPLE VALUE INPUT)	
QRESPM	RETURN AN ARRAY OF RESPONSES IN TEXT AND NUMERIC FORM, MAY INCLUDE INTEGER AND REAL VALUES
QRESPX	SAME AS QRESPM, INITIAL VALUES FROM TEXT STRING
QRESCN	SAME AS QRESPM BUT MAY HAVE MULTIPLE ROWS FOR 2-D
QRESCX	SAME AS QRESPX BUT MAY HAVE MULTIPLE ROWS FOR 2-D
QRESCD	SAME AS QRESCN BUT DOUBLE PRECISION ALLOWED
QRESCR	SAME AS QRESCX BUT DOUBLE PRECISION ALLOWED
QRESCZ	SAME AS QRESCR BUT WITH VARIABLE BUFFER LENGTH

Figure 4.9. One-dimensional array input subroutines.

Details of the arguments required in order for these routines to be called from an application program are found in Appendix B.1.

TWO-DIMENSIONAL Array Edit Subroutines

The 2-D data edit subroutines are used to prompt the user to enter numeric and/or text values in up to 30 fields and 49 rows of data. User input is checked against valid values and errors are reported if incorrect input is detected. Figure 4.10 contains a structure chart showing the relationships among the various two-dimensional data input subroutines. Note that these routines also may be used to prompt the user to enter one-dimensional data if the parameter "NROW" is set to 1. Figure 4.11 differentiates between the capabilities of the subroutines resident in this group.

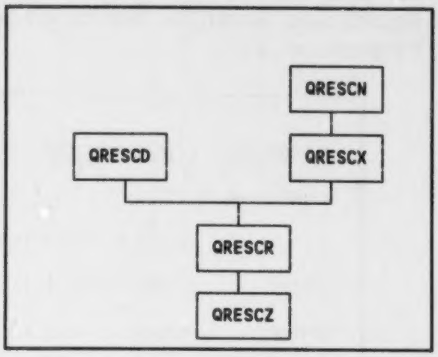


Figure 4.10. Structure chart for 2-D array edit subroutines.

SUBROUTINE	PRIMARY ACTIONS
(ALL REQUIRE SPECIFICATION OF NUMBER OF ROWS OF DATA)	
QRESCN	RETURN TEXT STRING OF RESPONSES ALONG WITH INTEGER AND REAL VALUES
QRESCX	SAME AS QRESCN BUT GETS INITIAL VALUES FROM TEXT STRING
QRESCD	SAME AS QRESCN BUT DOUBLE PRECISION ALLOWED
QRESCR	SAME AS QRESCX BUT DOUBLE PRECISION ALLOWED
QRESCZ	SAME AS QRESCR BUT WITH VARIABLE BUFFER LENGTH

Figure 4.11. Two-dimensional data edit subroutines.

Details of the arguments required in order for these routines to be called from an application program are found in Appendix B.1.

FILE Management Subroutines

FILE screens are used to enter names of files that are needed by an application. Figure 4.12 contains a structure chart showing the two file management subroutines. In Figure 4.13 the capabilities of the two routines are differentiated. Details of the arguments required in order for these two routines to be called from an application program are found in Appendix B.1.

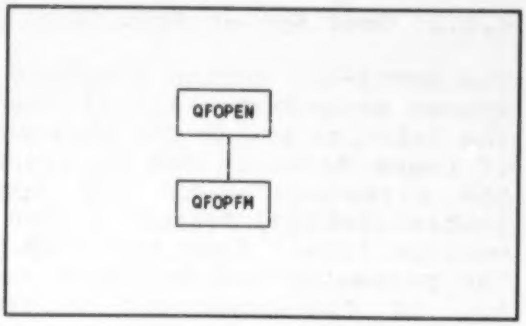


Figure 4.12. Structure chart for file subroutines.

SUBROUTINE	PRIMARY ACTIONS
QFOPEN	OPEN A FILE AND RETURN FORTRAN UNIT NUMBER, ASK USER TO CORRECT NAME IF ERROR OCCURS
QFOPFN	OPEN A FILE AND RETURN UNIT NUMBER, OPEN A FILE WITH SPECIFIED NAME, ASK USER TO CORRECT NAME IF ERROR OCCURS

Figure 4.13. File management subroutines.

4.2.8 Closing Comments on the Development Process

The most difficult part of the development process is developing the series of menu and data screens that allow the user to complete the input sequence in an efficient fashion. The user will be pleased with an interactive program if it allows entry of information in a familiar and logical manner while providing assistance where needed. It is most beneficial to sketch the planned sequence and detail of screens for end-user interactions at the beginning of the development effort.

Interactive program development is an iterative process. Users should test a prototype program at an early stage in the development process in order to identify additional needs with regards to screen sequence and content. Within the ANNIE-IDE framework the screen content can be changed quickly by editing the screen definition file and using the import function of MESSIE.

4.3 ADVANCED TOPICS

4.3.1 User System Specifications File (TERM.DAT)

The ANNIE-IDE system contains a number of parameters that may be system dependent. All of these parameters have a default value. The defaults are in the main message file, MESSAGE.WDM. Any number of these defaults can be overridden by adding a TERM.DAT file to the directory where the application is being executed. At initialization, default values for all parameters are read from the message file. Then the TERM.DAT file is read, if it exists, and the parameter values found replace those from the message file. All of the parameter values are saved in a common block. Application programmers can determine the value of a parameter in the common block by calling subroutine ANPRGT. Appendix B.6 lists the parameter code number (used only by the application programmer), parameter code name (needed for the users TERM.DAT file), default value, allowable values, and the definition for each parameter.

Figure 4.14 contains a sample user's TERM.DAT file. The first two parameters are the terminal input file (TRMINP) and terminal output file (TRMOUT). For a PC version of Fortran, the standard values for these parameters are five and six. The next parameter is the starting Fortran unit number to be available to the program. It is suggested this number be assigned a value of at least 40, as many unit numbers below that are used by ANNIE-IDE (e.g. WDM-message file). The fourth and fifth parameters are the computer type (CMPTYP) and terminal type (TRMTYP). For this example, the user is working on a PC, so the values of these parameters are set accordingly. The next three parameters are color specifications. The first color is the standard background color which has been set to one (blue). The next color is the foreground error color. This has been set to 15, which is bright white (as opposed to the standard foreground color default of light grey). The final color is the background error color. This has been set to 4, which is red. The error colors should be set to values that will draw attention to the error messages. Appendix B.8 contains descriptions of all allowable color codes. The final two parameters are function key definitions. Valid values for these parameters are the first letter of screen commands. Thus, the first parameter, function key 4 (PFKEY4), has been set to execute the 'CMNDS' command. The second parameter, function key 5 (PFKEY5), has been set to execute the 'LIMITS' command.

```
TRMINP 5
TRMOUT 6
FILUNI 40
CMPTYP PC
TRMTYP PC
CLRBKS 1
CLRFRE 15
CLRBKE 4
PFKEY4 C
PFKEY5 L
```

Figure 4.14. Sample TERM.DAT file.

4.3.2 Interscreen Movement Commands

The application programmer may make the interscreen movement commands described in Section 3.10 available through calls to toolkit routine ZSTCMA. The user's screen exit code is determined by the application program by a call to routine ZGTRET. Sections 5.13 and 5.15 show how this capability may be used in an application program.

4.3.3 Control of Assistance Window Containing System Status

The assistance window used to display system status information is controlled by the application programmer with calls to toolkit routine ZSTADD. This routine updates a record within the 10 record status buffer. An example of how this function is used may be found in Section 5.12.

SECTION 5

EXAMPLE APPLICATIONS

This section provides examples and supporting text illustrating the use of ANNIE-IDE software to build the components of an interactive program. As appropriate, each example consists of one or more of the following: the Fortran code needed to perform the function, the screen directive needed to define screen display, and the resulting screen.

It should be noted that the examples utilize certain subordinate subroutines that are discussed in Section 4.2. Details for these subroutines can be found in Appendix B.1. Screen directives used in the examples are explained in Section 4.2 and summarized in Appendix B.4.

The examples in this section appear in the general order in which they might occur within an interactive program. That is to say, the examples begin with initializing the system and end with closing the system. Further details on sequencing the functions of an interactive program are available in Section 4.2.

5.1 INITIALIZE SYSTEM

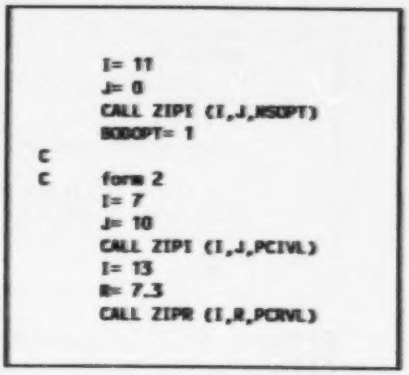
The toolkit housekeeping routine ANINIT is responsible for initializing ANNIE-IDE for each session. It opens the WDM-message file and sets system parameters (either by default or by user specifications in TERM.DAT). The Fortran unit number of the WDM-message file (MESSFL) is required as an input argument. This call must be made before any other ANNIE-IDS routines are called. Figure 5.1 shows sample code to perform this task. This code was taken from the DBAPE main program. The application programmer must select a Fortran unit number for the WDM-message file that does not conflict with others assigned by ANNIE-IDE through the TERM.DAT parameter FILUNI.

```
C
C     initialize ANNIE-IDE
      MESSFL= 30
      CALL ANINIT (MESSFL)
C
```

Figure 5.1. Fortran code to initialize ANNIE-IDE.

5.2 INITIALIZE MODEL PARAMETERS

The code in Figure 5.2 shows examples of data initialization required to set default values for an environmental model. The main initialization routines are ZIPC (character), ZIPI (integer), ZIPR (real), and ZIPD (double precision). These routines are particularly helpful in initializing data arrays. The first argument (I) is the number of elements in the array being initialized. The second argument (J or R) is the initial value desired. The third argument is the array being initialized. This code comes from ANNIE-QUAL2 code group ANNSQ2M, routine AQ2INI. This form of initialization (setting default values for parameters in the individual data entry screens) is required because the user may choose not to enter data in each of the individual data screens, thus failing to initialize some parameter values.



```
      I= 11
      J= 0
      CALL ZIPI (I,J,NSOPT)
      BODOPT= 1
C
C     form 2
      I= 7
      J= 10
      CALL ZIPI (I,J,PCIVL)
      I= 13
      R= 7.3
      CALL ZIPR (I,R,PCRVL)
```

Figure 5.2. Fortran code to initialize model parameters.

5.3 READ FILE INPUT FOR A MODEL

The code in Figure 5.3 shows a method for reading input for a model from a file. In this example, the program reads a formatted sequential file containing the input sequence for a model; reading continues until a keyword is reached (or until a boundary is reached). Thus, for each set of parameters, reading continues until the temporary string, TCHR, matches the keyword, ENDA. This is checked by the function STRFND, which searches the first string (TCHR) looking for the second string (ENDA). ENDA is defined with a Fortran DATA statement. The argument LEN is the length of each string. Often the lengths are two different arguments with the first larger than the second. Each loop is completed when either STRFND has a value greater than 1 or the limit of the array sizes for the parameters is reached. If a read error occurs on a read from the file, then control is passed to label 900 where a message is printed to the screen indicating the nature of the error. The code comes from ANNIE-QUAL2 code group ANNSQ2M, subroutine AQ2REA.

```
C
C     read form 5
      RETCOD= 5
      I     = 0
 50   CONTINUE
        I= I+ 1
        READ (SUCIFL,1050,ERR=900) TCHR,DAT2(1,I),(RCHNAM(J,I),J=1,16),
     1                             DAT2(2,I),DAT2(3,I)
        J= STRFND(LEN,TCHR,LEN,ENDA)
      IF (I.LT.MXRCH.AND.J.EQ.0) GO TO 50
      IF (J.EQ.0) READ (SUCIFL,1000,ERR=900)
C
```

Figure 5.3. Fortran code to read model parameters from a file.

5.4 DISPLAY TEXT

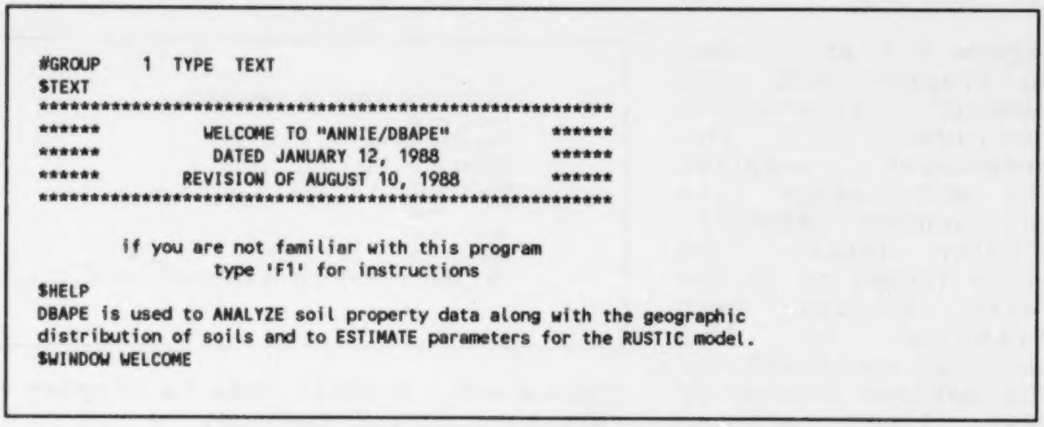
Figure 5.4 is an example of code required to display on the screen the text stored in the WDM-message file. The programmer supplies the WDM-message file unit number (MESSFL), the screen cluster number (SCLU), and the screen group number (SGRP) within the cluster. The subroutine PRNTXT is the simplest text-displaying routine. It initializes the data screen to blanks, displays the text and instructions, and waits for the user to continue to the next screen. This code is from the DBAPE main program.

```
C
C     initialize screen cluster number
      SCLU= 36
C     welcome message
      SGRP= 1
      CALL PRNTXT (MESSFL,SCLU,SGRP)
C
```

Figure 5.4. Fortran code to display text.

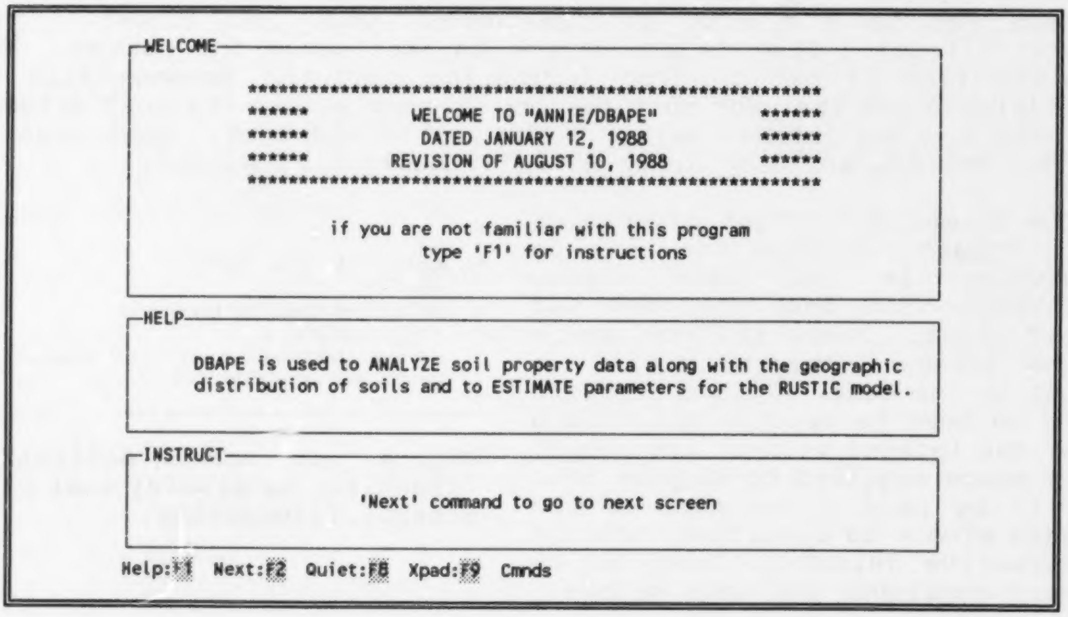
The screen definition directives in Figure 5.5 contain instructions to display text ($TEXT), provide associated help information ($HELP) and define the name that labels the data window ($WINDOW). Both the text and help are automatically centered within their respective windows when they are displayed.

Figure 5.6 shows the text screen that is displayed based on the code and screen definition directives shown (after the user has asked for the display of help information).



```
#GROUP    1  TYPE  TEXT
$TEXT
**********************************************************
******          WELCOME TO "ANNIE/DBAPE"          ******
******           DATED JANUARY 12, 1988           ******
******        REVISION OF AUGUST 10, 1988         ******
**********************************************************

        if you are not familiar with this program
                type 'F1' for instructions
$HELP
DBAPE is used to ANALYZE soil property data along with the geographic
distribution of soils and to ESTIMATE parameters for the RUSTIC model.
$WINDOW WELCOME
```

Figure 5.5. Screen definition directives to display text.



```
-WELCOME-
          **********************************************************
          ******          WELCOME TO "ANNIE/DBAPE"          ******
          ******           DATED JANUARY 12, 1988           ******
          ******        REVISION OF AUGUST 10, 1988         ******
          **********************************************************

                  if you are not familiar with this program
                          type 'F1' for instructions

-HELP-
     DBAPE is used to ANALYZE soil property data along with the geographic
     distribution of soils and to ESTIMATE parameters for the RUSTIC model.

-INSTRUCT-
                    'Next' command to go to next screen

Help:F1  Next:F2  Quiet:F8  Xpad:F9  Cmnds
```

Figure 5.6. Resulting text screen.

5.5 DISPLAY TEXT WITH NUMERIC INFORMATION

Figure 5.7 shows code to display text with numeric information included. The programmer supplies the WDM-message file unit number (MESSFL), cluster (SCLU), and group (SGRP) as in the basic display text example in the previous section(5.4). The maximum number of text lines allowed (MXLIN), a screen initialization flag (SCNINI), a write flag (IO), the number of parameters (I), and the value of the parameters (IVAL) also must be specified. In this example, the maximum number of lines has been set to the size of the screen text (50 lines). The initialization flag is set to one to initialize the screen. The write flag is set to zero indicating that the message will be displayed and the user must respond to continue to the next screen. There are two integer values to be put in the text. This code is from ANNIE-QUAL2 code group ANNSQ2S, subroutine AQUAL2.

```
C       conflicting number of headwaters
        IVAL(1)= J
        IVAL(2)= NUMHWD
        SCNINI = 1
        MXLIN  = 50
        I      = 2
        IO     = 0
        SGRP   = 5
        CALL PMXTXI (MESSFL,SCLU,SGRP,MXLIN,SCNINI,IO,I,IVAL)
```

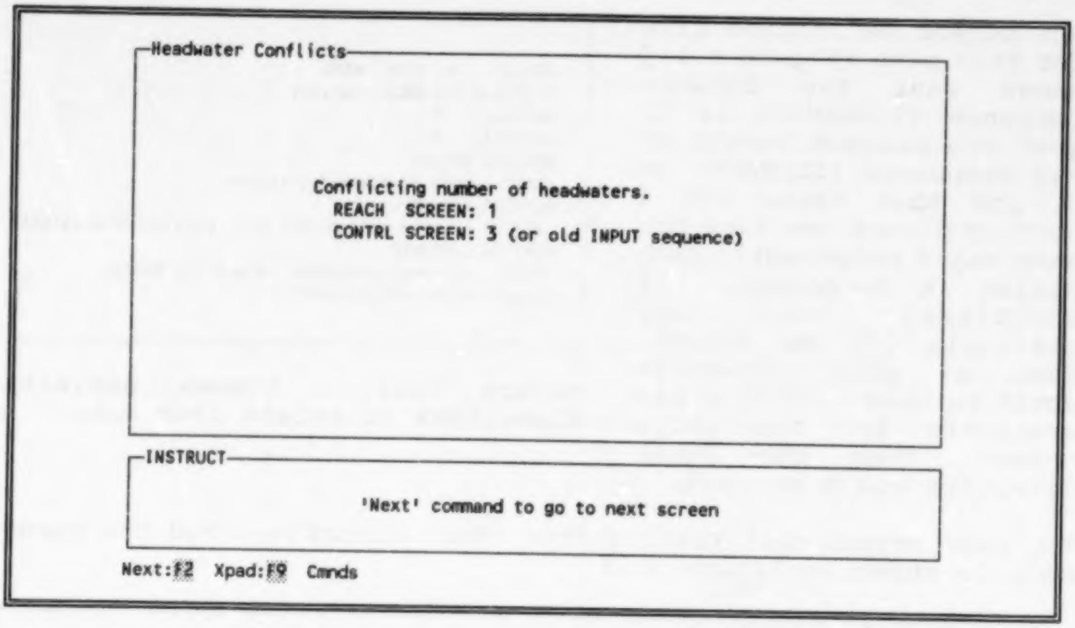
Figure 5.7. Fortran code to display text with numeric information.

The screen definition directives in Figure 5.8 show the text to display in the data window ($TEXT). The text includes two "&" signs. These indicate where the integer parameters will be put in the text. Note that there is no need to specify the length of the integer values; the amount of space required to display them will be used. The name of the data window is specified with the directive $WINDOW. There is no help available for this screen.

```
#GROUP    5  TYPE  TEXT
$TEXT
Conflicting number of headwaters.
  REACH  SCREEN: &
  CONTRL SCREEN: &  (or old INPUT sequence)
$WINDOW Headwater Conflicts
```

Figure 5.8. Screen definition directives to display text with numeric information.

Figure 5.9 shows the text screen that results from the code and screen directives just shown.



```
-Headwater Conflicts-

        Conflicting number of headwaters.
          REACH  SCREEN: 1
          CONTRL SCREEN: 3 (or old INPUT sequence)

-INSTRUCT-
             'Next' command to go to next screen
Next:F2  Xpad:F9  Cmnds
```

Figure 5.9. Resulting text screen with numeric information.

5.6 SELECT AN ITEM FROM A MENU

Shown in Figure 5.10 is the code required to prompt the user to select an item from a menu. The programmer supplies the WDM-message file unit number (MESSFL), the screen cluster number (SCLU), and the screen group number (SGRP) within the cluster to subroutine QRESP. The order number of the selected response (RESP) is returned. The program then branches based on the response. This example is from the DBAPE main program.

```
C       main loop for DBAPE, which option
        SGRP= 6
        CALL QRESP (MESSFL,SCLU,SGRP,RESP)
        GO TO (10,20,30), RESP
```

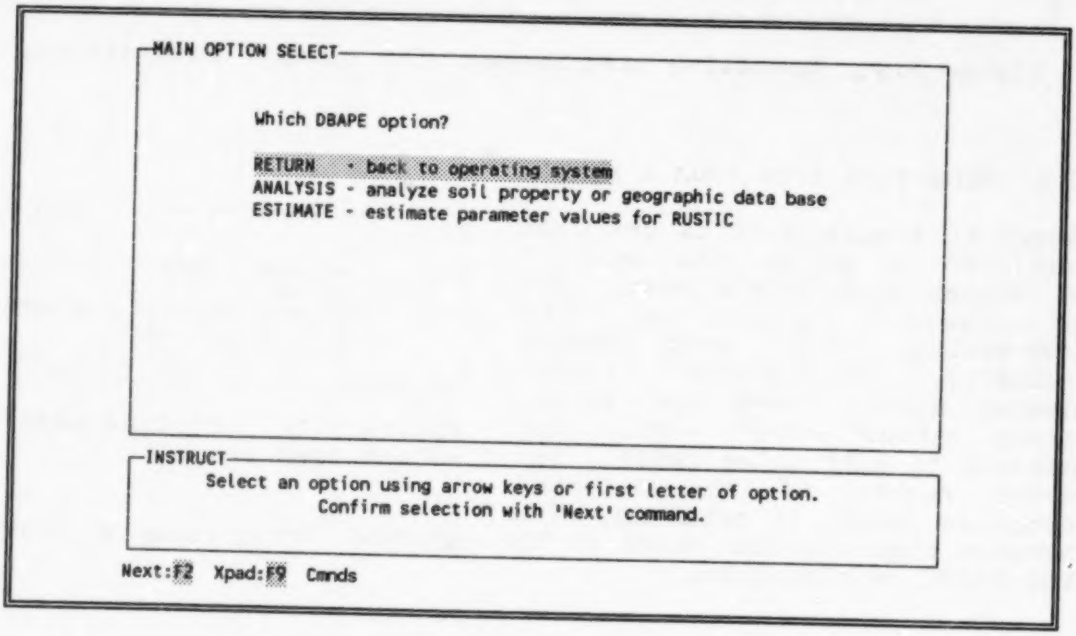
Figure 5.10. Fortran code to select from menu.

The screen definition file for this menu (Figure 5.11) shows that the default response ($DEFAULT) is 1, that the maximum length of the responses ($LENGTH) is 8, and that there are 3 $OPTION directives (one for each valid response). Each option is described. If additional help was available for an option, then a _HELP directive would be used. If help was available for the whole screen, then the $HELP directive would be used.

```
#GROUP    6  TYPE  MENU
$TITLE Which DBAPE option?
$DEFAULT   1
$LENGTH    8
$OPTION RETURN
_DESC - back to operating system
$OPTION ANALYSIS
_DESC - analyze soil property or geographic data base
$OPTION ESTIMATE
_DESC - estimate parameter values for RUSTIC
$WINDOW MAIN OPTION SELECT
```

Figure 5.11. Screen definition directives to select from menu.

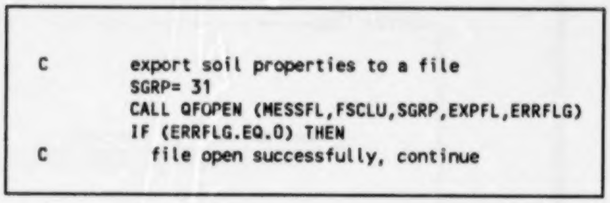
The menu screen that results from these directives and the Fortran code is shown in Figure 5.12.



```
MAIN OPTION SELECT

        Which DBAPE option?

        RETURN   - back to operating system
        ANALYSIS - analyze soil property or geographic data base
        ESTIMATE - estimate parameter values for RUSTIC

INSTRUCT
      Select an option using arrow keys or first letter of option.
              Confirm selection with 'Next' command.

Next:F2  Xpad:F9  Cmnds
```

Figure 5.12. Resulting menu screen.

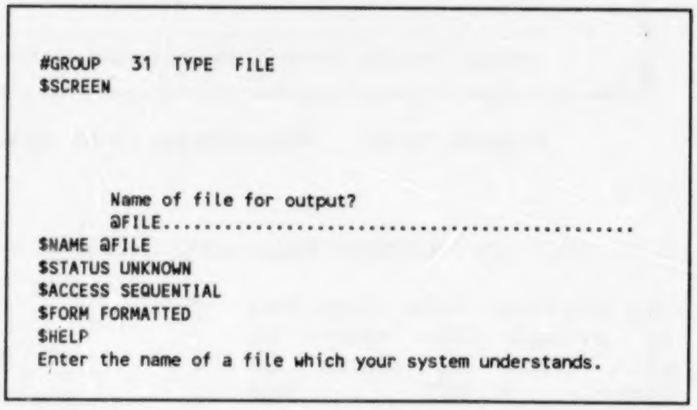
5.7 OPEN A FILE



```
C       export soil properties to a file
        SGRP= 31
        CALL QFOPEN (MESSFL,FSCLU,SGRP,EXPFL,ERRFLG)
        IF (ERRFLG.EQ.0) THEN
C          file open successfully, continue
```

Figure 5.13. Fortran code to specify a file.

Figure 5.13 contains code to display a screen that asks the user for a file name and opens the file. The message file (MESSFL), cluster (FSCLU), and group (SGRP) are supplied by the programmer. The routine returns the unit number of the file that was just opened (EXPFL). The final argument (ERRFLC) indicates whether the opening of the file was successful. Any non-zero value for this argument indicates an error occurred. If no error has occurred, then the program proceeds to write to the file (not shown here). This example is from DBAPE code group SOIMAK, subroutine SOIMAK.

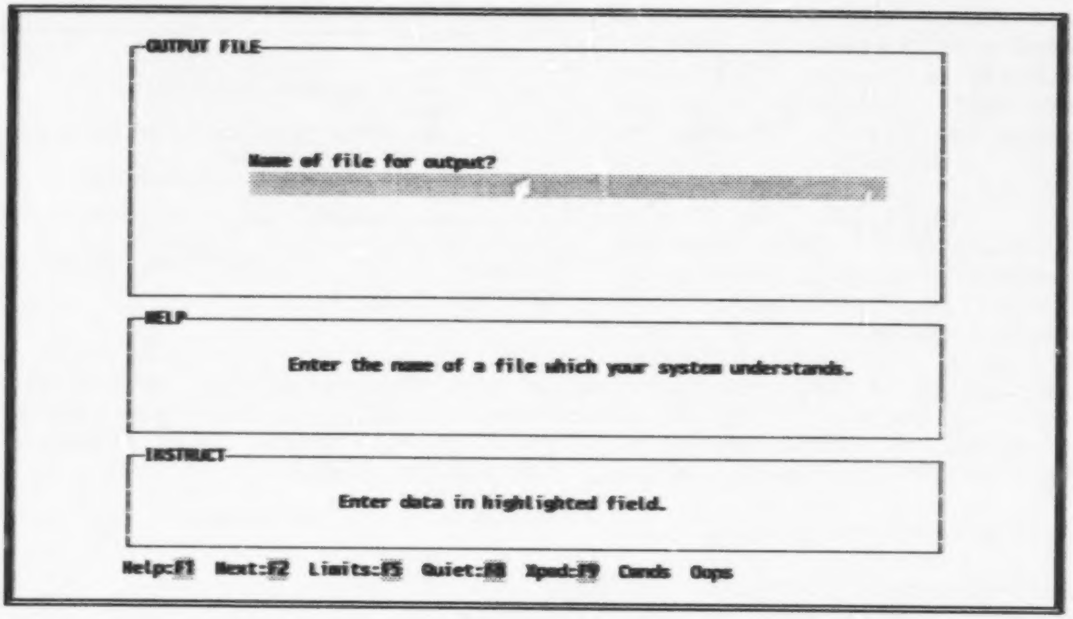


```
#GROUP   31  TYPE  FILE
$SCREEN



         Name of file for output?
         @FILE..............................................
$NAME @FILE
$STATUS UNKNOWN
$ACCESS SEQUENTIAL
$FORM FORMATTED
$HELP
Enter the name of a file which your system understands.
```

Figure 5.14. Screen definition directives to specify a file.

The screen definition in Figure 5.14 demonstrates that the $SCREEN directive is used here in the same manner as in the PRM1 type screen. The field in which the user enters the file name is set using the "@FILE" text with a string of "."s following it. A length of 64 is suggested for this field, so that the user may enter long file names if necessary. The $NAME directive is similar here to the $FIELD directive in the PRM1 type screen and is thus followed with the field name "@FILE," which appears in the screen text. The directives $STATUS, $ACCESS, and $FORM specify the file as being sequential, formatted, and of unknown status. Help to the user is provided through the $HELP directive.

The file specification screen that results from these directives and code (after the user has requested help information) is shown in Figure 5.15.



OUTPUT FILE

Name of file for output?

HELP

Enter the name of a file which your system understands.

INSTRUCT

Enter data in highlighted field.

Figure 5.15. Resulting file specification screen.

5.8 EDIT ONE-DIMENSIONAL ARRAY OF DATA

The Fortran code required to prompt the user to edit data is shown in Figure 5.16. The programmer supplies the message file (MESSFL), cluster (SCLU), group (SGRP), number of integer values (I1), number of real values (FNUM), number of character values (I1), number of rows of data (set to 1 for one-dimensional data), and a flag, which is set to 1 for this application. The routine QRESCN modifies the integer, real, and character arrays (IVAL,RVAL,CVAL) as well as the character*1 buffer containing the data (TBUFF, dimensioned (80)). The arrays all must be dimensioned by a value of at least one, even if there are no data values for the respective array. In this case, the real array is dimensioned

```
C        get min, max, and include info
         I1  = 1
         SGRP= 26
         FNUM= 2
         RVAL(1)= 10
         RVAL(2)= 10
         CVAL(1,1)= 10
         CALL QRESCN (MESSFL,SCLU,SGRP,I1,FNUM,I1,I1,I1,
  M                   IVAL,RVAL,CVAL,TBUFF)
         SOFMIN(SOFCNT)= RVAL(1)
         SOFMAX(SOFCNT)= RVAL(2)
         SOFCND(SOFCNT)= CVAL(1,1)- 1
```

Figure 5.16. Fortran code to enter 1-D data.

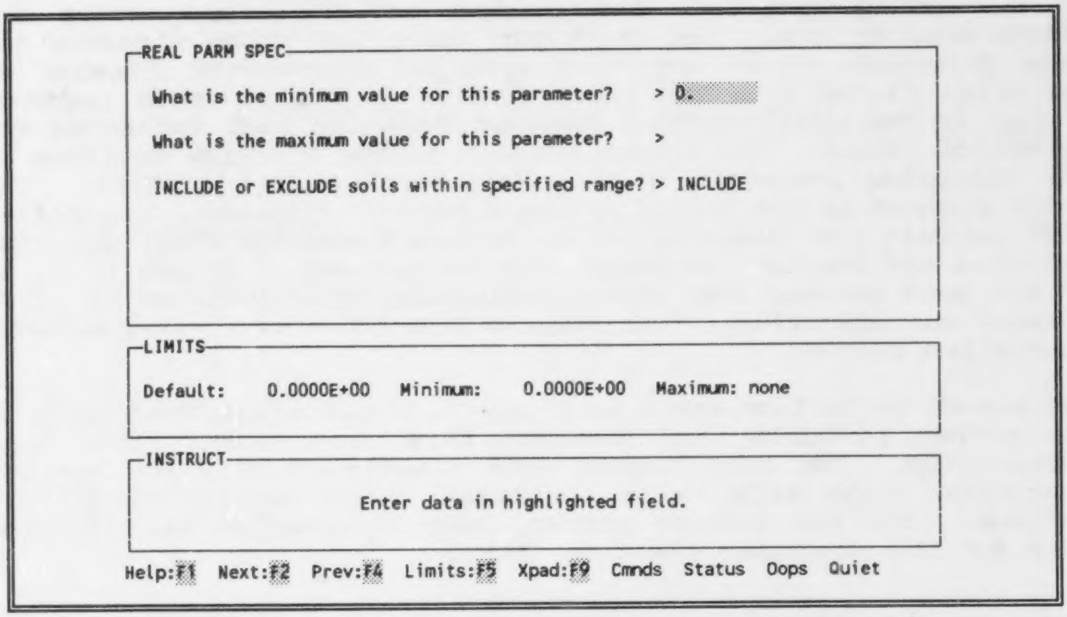
by two (FNUM) and both the integer and character arrays are dimensioned by one. The character array has three elements for each character field and thus must be dimensioned (number of character fields,3). The first element is the response sequence number in the user-selected response from the list following the directive _VALID. The second element is the starting position of the character response in the character*1 buffer (TBUFF). The third element is the length of the character response. The buffer will contain the modified array values based on their starting position and length. Although the integer array is set to 1, it is not used because the screen definition file contains no _TYPE integer specification. This code is from DBAPE code group SOIMAK, subroutine SOIFSP.

The screen definition shown in Figure 5.17 specifies the format of the screen ($SCREEN) and provides first and second level help information. Two real fields with ranges and defaults and one character field with valid responses and a default value are defined. For the defined screen, help information is available only for the character field for PRM3.

```
#GROUP    26   TYPE   PRM1
$SCREEN

  What is the minimum value for this parameter?     > @PRM1.....

  What is the maximum value for this parameter?     > @PRM2.....

  INCLUDE or EXCLUDE soils within specified range? > @PRM3..

$FIELD @PRM1
_TYPE REAL
_RANGE       0.000:  -999.000
_DEFAULT     0.000
$FIELD @PRM2
_TYPE REAL
_RANGE       0.000:  -999.000
_DEFAULT     0.000
$FIELD @PRM3
_TYPE CHARACTER
_VALID INCLUDE, EXCLUDE
_DEFAULT INCLUDE
_HELP
If you want to find soils within the specified numeric range, enter INCLUDE.
If soils not in the specified range are desired, enter EXCLUDE.
$WINDOW REAL PARM SPEC
```

Figure 5.17. Screen definition directives to enter 1-D data.

The one-dimensional data entry screen resulting from these directives and the Fortran code (after the user has requested limits information for the first field) is shown in Figure 5.18.



```
REAL PARM SPEC
  What is the minimum value for this parameter?    > 0.

  What is the maximum value for this parameter?    >

  INCLUDE or EXCLUDE soils within specified range? > INCLUDE

LIMITS
 Default:     0.0000E+00   Minimum:     0.0000E+00   Maximum: none

INSTRUCT
                  Enter data in highlighted field.

Help:F1  Next:F2  Prev:F4  Limits:F5  Xpad:F9  Cmnds  Status  Oops  Quiet
```

Figure 5.18. Resulting 1-D data screen.

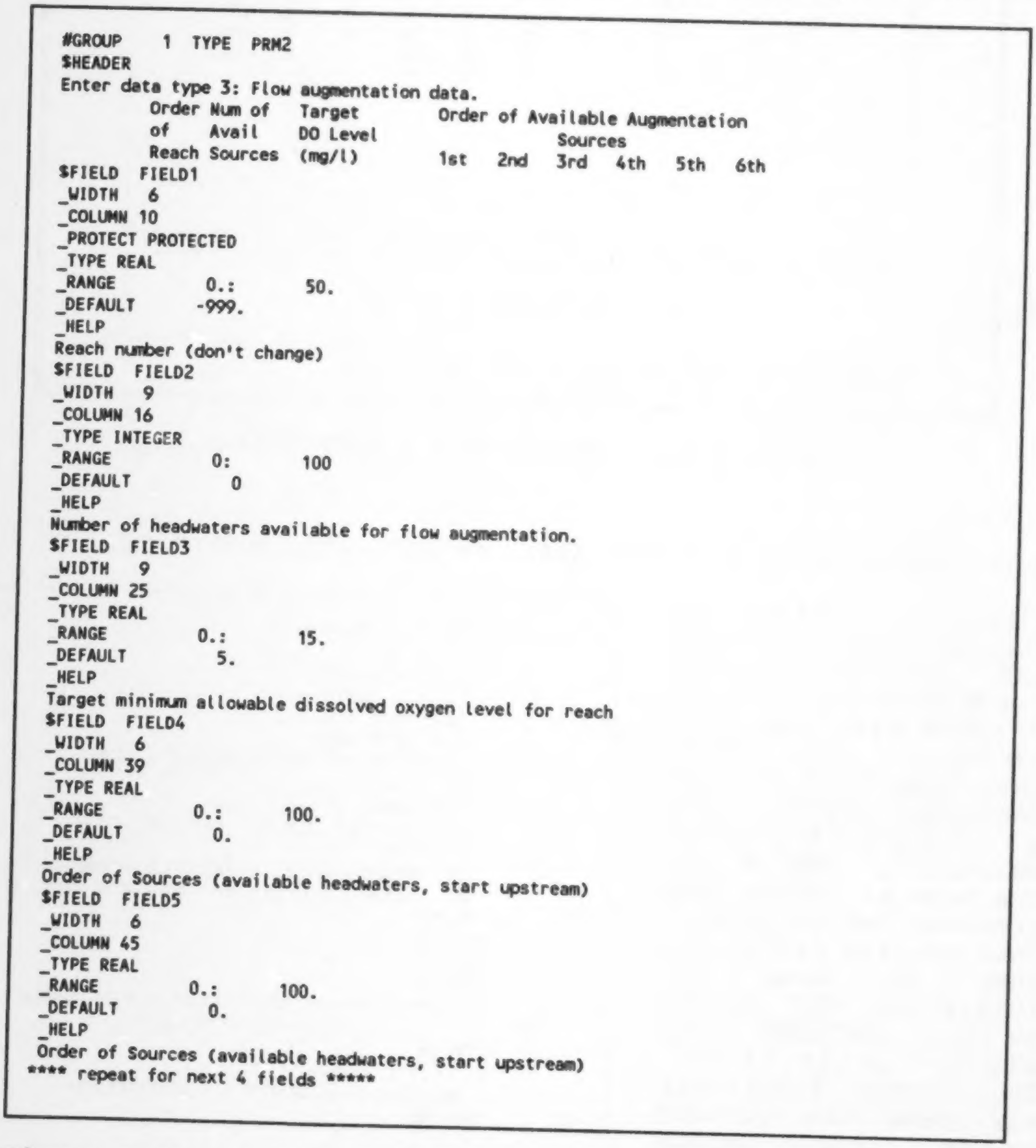
5.9 EDIT TWO-DIMENSIONAL ARRAY OF DATA

```
C          group data3, form 6
           I1= 1
           DO 65 J= 1, NUMRCH
             DAT3R(1,J)= DAT2(1,J)
 65        CONTINUE
           SGRP= 13
           RNUM = 8
           CALL QRESCN (MESSFL,SGRP,I1,I1,RNUM,I1,NUMRCH,I1,
     M                  DAT3I,DAT3R,CVAL,SCHRBF)
```

Figure 5.19. Fortran code to enter 2-D data.

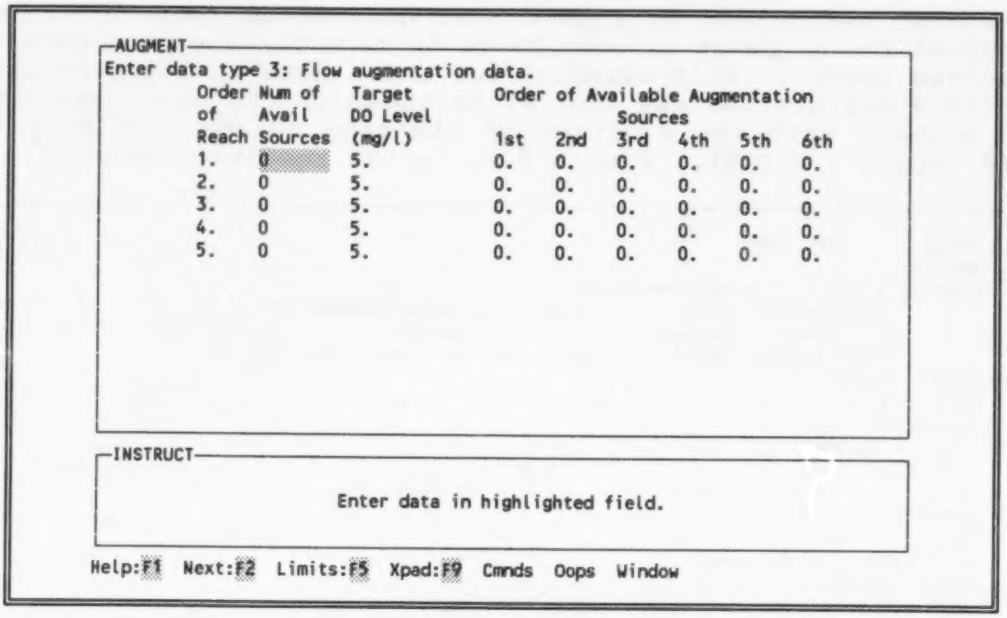
The arguments for this example are identical to those in the example in Section 5.8. The main difference between the two is that the arrays now have two dimensions. The second dimension is the number of rows of data, here defined by the variable NUMRCH (which was defined earlier by the program). Thus, all the arrays must now be dimensioned by the number of integer, real, or character fields and by the maximum number of data rows possible. The character array must now be dimensioned (number of character fields,3,number of data rows). In this example, the integer (DAT3I) and real (DAT3R) values being modified are already arrays dimensioned to (1,NUMRCH) and (RNUM,NUMRCH). This code is from ANNIE-QUAL2 code group ANNSQ2S, subroutine AQUAL2.

The screen definition in Figure 5.20 designates field 1 to be a protected field, as it is only there to show the user the data row (a stream reach in this example) for which data is being entered. The field definitions are similar to those of the one-dimensional data screen, with the addition of starting column (_COLUMN) and field length (_WIDTH). Figure 5.21 is the resulting screen.



```
#GROUP    1  TYPE  PRM2
$HEADER
Enter data type 3: Flow augmentation data.
         Order Num of  Target         Order of Available Augmentation
         of    Avail   DO Level                   Sources
         Reach Sources (mg/l)         1st   2nd   3rd   4th   5th   6th
$FIELD  FIELD1
_WIDTH   6
_COLUMN 10
_PROTECT PROTECTED
_TYPE REAL
_RANGE        0.:       50.
_DEFAULT      -999.
_HELP
Reach number (don't change)
$FIELD  FIELD2
_WIDTH   9
_COLUMN 16
_TYPE INTEGER
_RANGE        0:        100
_DEFAULT        0
_HELP
Number of headwaters available for flow augmentation.
$FIELD  FIELD3
_WIDTH   9
_COLUMN 25
_TYPE REAL
_RANGE        0.:       15.
_DEFAULT        5.
_HELP
Target minimum allowable dissolved oxygen level for reach
$FIELD  FIELD4
_WIDTH   6
_COLUMN 39
_TYPE REAL
_RANGE        0.:       100.
_DEFAULT        0.
_HELP
Order of Sources (available headwaters, start upstream)
$FIELD  FIELD5
_WIDTH   6
_COLUMN 45
_TYPE REAL
_RANGE        0.:       100.
_DEFAULT        0.
_HELP
Order of Sources (available headwaters, start upstream)
**** repeat for next 4 fields *****
```

Figure 5.20. Screen definition directives to enter 2-D data.



```
AUGMENT
Enter data type 3: Flow augmentation data.
          Order Num of  Target          Order of Available Augmentation
          of    Avail   DO Level                    Sources
          Reach Sources (mg/l)        1st   2nd   3rd   4th   5th   6th
          1.    0       5.            0.    0.    0.    0.    0.    0.
          2.    0       5.            0.    0.    0.    0.    0.    0.
          3.    0       5.            0.    0.    0.    0.    0.    0.
          4.    0       5.            0.    0.    0.    0.    0.    0.
          5.    0       5.            0.    0.    0.    0.    0.    0.

INSTRUCT
                    Enter data in highlighted field.

Help:F1  Next:F2  Limits:F5  Xpad:F9  Cmnds  Oops  Window
```

Figure 5.21. Resulting 2-D data screen.

5.10 ENTER TWO-DIMENSIONAL ARRAY OF DATA WITH BUFFER USE

Figure 5.22 shows the Fortran code to prompt the user to enter a two-dimensional array of data with use of a buffer by the programmer to set character fields. The arguments for subroutine QRESCX are the same as QRESCN (see previous two examples). This routine differs in that it does not initialize the screen buffer (SCHRBF) to blank characters. Thus, the programmer may pass the current values into QRESCX through the buffer. This is necessary because field two is an

```
C       group data2, form 5
        I1= 1
C       initialize screen
        DO 52 I= 1, NUMRCH
          LEN  = 6
C         reach order number
          CALL DECCHR (DAT2(1,I),LEN,JUST,K,SCHRBF(1,I))
          LEN  = 16
C         reach name
          CALL CHRCHR(LEN,RCHNAM(1,I),SCHRBF(7,I))
          LEN  = 10
          CALL DECCHR (DAT2(2,I),LEN,JUST,K,SCHRBF(23,I))
          CALL DECCHR (DAT2(3,I),LEN,JUST,K,SCHRBF(33,I))
 52     CONTINUE
        SCLU= 12
        SGRP= 1
        RNUM= 3
        CALL QRESCX (MESSFL,SCLU,SGRP,I1,RNUM,I1,NUMRCH,I1,
     M               IVAL,DAT2,CVAL,SCHRBF)
        LEN= 16
        DO 53 I= 1, NUMRCH
          CALL CHRCHR(LEN,SCHRBF(7,I),RCHNAM(1,I))
 53     CONTINUE
```

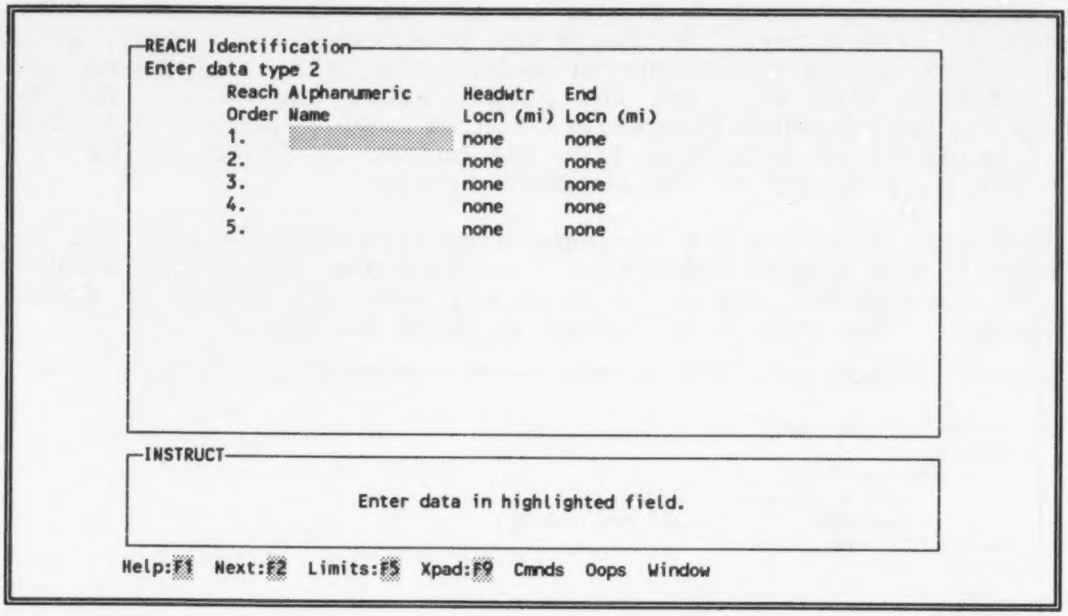
Figure 5.22. Fortran code to enter 2-D data using buffer.

alphanumeric name and thus has no set of valid responses as is needed by CVAL arrays. Although the programmer has put all of the data array values in the buffer before the call, he does not need to retrieve them all from the buffer after the call. The data array values returned from QRESCX can be used. The only data field that needs to be retrieved from the buffer is field two as it is not returned in any of the argument arrays.

Figure 5.23 contains the corresponding screen definition for this example. The screen definition for field two shows that although it is a character field, it does not have any valid or default responses. The resulting screen is shown in Figure 5.24.

```
#GROUP    1  TYPE  PRM2
$HEADER
Enter data type 2
        Reach Alphanumeric    Headwtr   End
        Order Name            Locn (mi) Locn (mi)
$WINDOW REACH Identification
$FIELD   FIELD1
_WIDTH   6
_COLUMN  10
_PROTECT PROTECTED
_TYPE    REAL
_RANGE   0. : 50.
_DEFAULT -999.
_HELP
Reach number
$FIELD   FIELD2
_WIDTH   16
_COLUMN  16
_TYPE    CHARACTER
_HELP
Name of reach
$FIELD   FIELD3
_WIDTH   10
_COLUMN  32
_TYPE    REAL
_RANGE   0. : -999.
_DEFAULT -999.
_HELP
River mile at head of reach
$FIELD   FIELD4
_WIDTH   10
_COLUMN  42
_TYPE    REAL
_RANGE   0. : -999.
_DEFAULT 0.
_HELP
River mile at end of reach
$HELP
See instructions for form 5 for more information.
```

Figure 5.23. Screen definition directives to enter 2-D data with use of buffer.

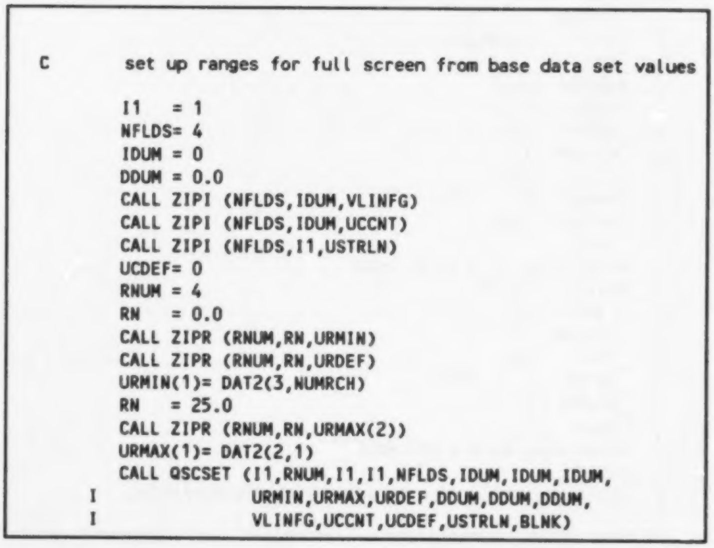


```
REACH Identification
Enter data type 2
          Reach Alphanumeric     Headwtr   End
          Order Name             Locn (mi) Locn (mi)
          1.                     none      none
          2.                     none      none
          3.                     none      none
          4.                     none      none
          5.                     none      none

INSTRUCT
                    Enter data in highlighted field.

Help:F1  Next:F2  Limits:F5  Xpad:F9  Cmnds  Oops  Window
```

Figure 5.24. Resulting 2-D data screen with use of buffer.

5.11 SET LIMITS AND DEFAULTS FOR ONE OR TWO-DIMENSIONAL DATA

Figure 5.25 shows code used to set up a call to QSCSET. This routine allows the programmer to set limits and defaults for one- or two-dimensional data that will override the limits and default on the screen definition file. The first four arguments supply the number of fields for each type of data (integer, real, double precision, character). The fifth argument is the total number of fields for the current screen. For



```
C       set up ranges for full screen from base data set values

        I1   = 1
        NFLDS= 4
        IDUM = 0
        DDUM = 0.0
        CALL ZIPI (NFLDS,IDUM,VLINFG)
        CALL ZIPI (NFLDS,IDUM,UCCNT)
        CALL ZIPI (NFLDS,I1,USTRLN)
        UCDEF= 0
        RNUM = 4
        RN   = 0.0
        CALL ZIPR (RNUM,RN,URMIN)
        CALL ZIPR (RNUM,RN,URDEF)
        URMIN(1)= DAT2(3,NUMRCH)
        RN   = 25.0
        CALL ZIPR (RNUM,RN,URMAX(2))
        URMAX(1)= DAT2(2,1)
        CALL QSCSET (I1,RNUM,I1,I1,NFLDS,IDUM,IDUM,IDUM,
     I               URMIN,URMAX,URDEF,DDUM,DDUM,DDUM,
     I               VLINFG,UCCNT,UCDEF,USTRLN,BLNK)
```

Figure 5.25. Fortran code to set limits for data fields.

each type of numeric data, the minimum, maximum, and default values must be specified (arguments 6 through 14). The argument VLINFG is an array which specifies whether there are valid (set to 1) or invalid (set to 2) values for each field. If the value for a field is set to zero, there are no valid or invalid values for this field. The argument UCCNT is an array of the number of valid or invalid values specified for each field (set to zero if previous argument is zero). The argument UCDEF is the default response for each character field. The argument USTRLN specifies the length of valid or invalid responses (if they exist) for each field. The final argument is a buffer containing the valid or invalid responses. In this example, the data fields are all real fields. Thus, for all other data types, dummy values are used.

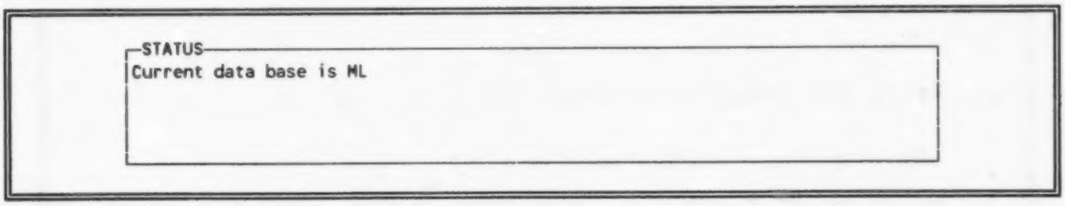
5.12 DISPLAY SYSTEM STATUS INFORMATION

Figure 5.26 contains code to place information in the first two lines of the status display. The programmer supplies the line number in the status text (1-10) and the text to display (TXTSTR). The text is any length up to 78. The status box is automatically displayed in the assistance box of the next screen that appears.

```
C       put current data base in status
        TXTSTR= 'Current data base is '//CDBTYP
        CALL ZSTADD(I1,TXTSTR)
        TXTSTR= ' '
        I= 2
        CALL ZSTADD(I,TXTSTR)
```

Figure 5.26. Fortran code to add status information.

Figure 5.27 shows the assistance box that results from the code just discussed.

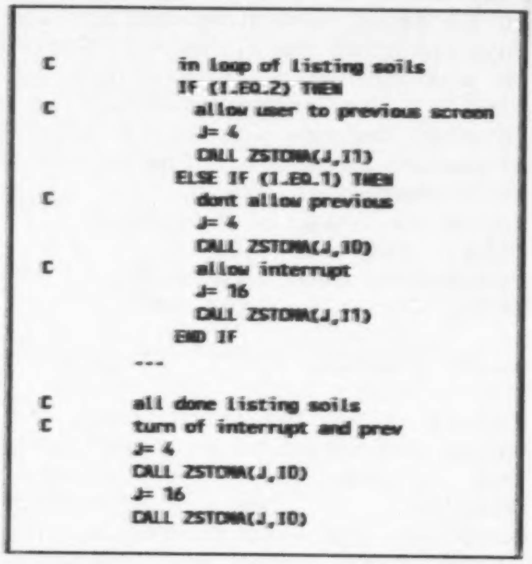


Current data base is ML

Figure 5.27. Resulting status display.

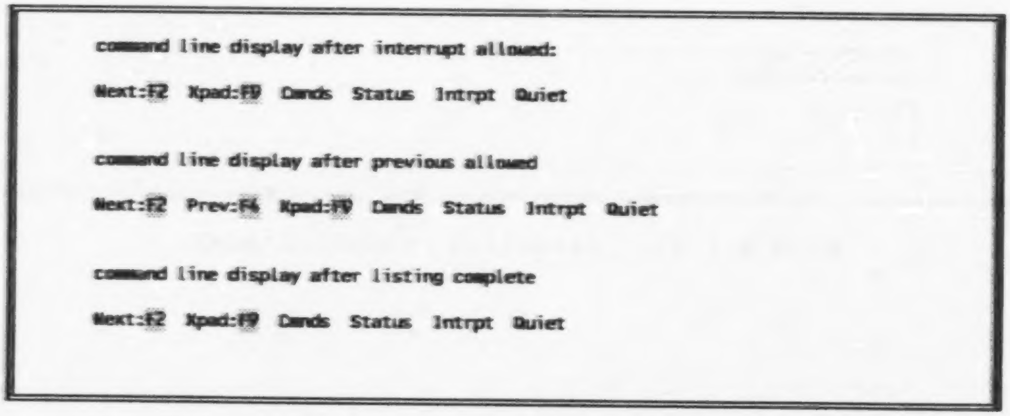
5.13 SET COMMAND AVAILABILITY

Figure 5.28 contains Fortran code to make commands appear and/or disappear on the command line. The programmer supplies the command number (J) to be made available or unavailable. The second argument is a flag to make the command available (set to 1) or unavailable (set to 0). A list of the command numbers is found in Appendix B.7. In this example, a series of listings are being displayed. When the first listing is displayed (I equals 1 conditional), the user is allowed to interrupt the listing, thus the interrupt command is made available (see Figure 5.29). When the user progresses to the second listing (I equals 2 conditional), the user is allowed to back through the listings, thus the previous command is made available (see Figure 5.29). If the user backs up to the first listing, the previous command would be made unavailable again. Once the user has finished viewing the listings, the interrupt and previous commands are made unavailable (see Figure 5.29). This code is from DBAPE code group SOIMAK, subroutine SOIPRT.



```
C           in loop of listing soils
            IF (I.EQ.2) THEN
C             allow user to previous screen
              J= 4
              CALL ZSTCMA(J,I1)
            ELSE IF (I.EQ.1) THEN
C             dont allow previous
              J= 4
              CALL ZSTCMA(J,I0)
C             allow interrupt
              J= 16
              CALL ZSTCMA(J,I1)
            END IF
        ---

C       all done listing soils
C       turn of interrupt and prev
        J= 4
        CALL ZSTCMA(J,I0)
        J= 16
        CALL ZSTCMA(J,I0)
```

Figure 5.28. Fortran code to set command availability.



```
command line display after interrupt allowed:

Next:F2  Xpad:F9  Cmnds  Status  Intrpt  Quiet

command line display after previous allowed

Next:F2  Prev:F4  Xpad:F9  Cmnds  Status  Intrpt  Quiet

command line display after listing complete

Next:F2  Xpad:F9  Cmnds  Status  Intrpt  Quiet
```

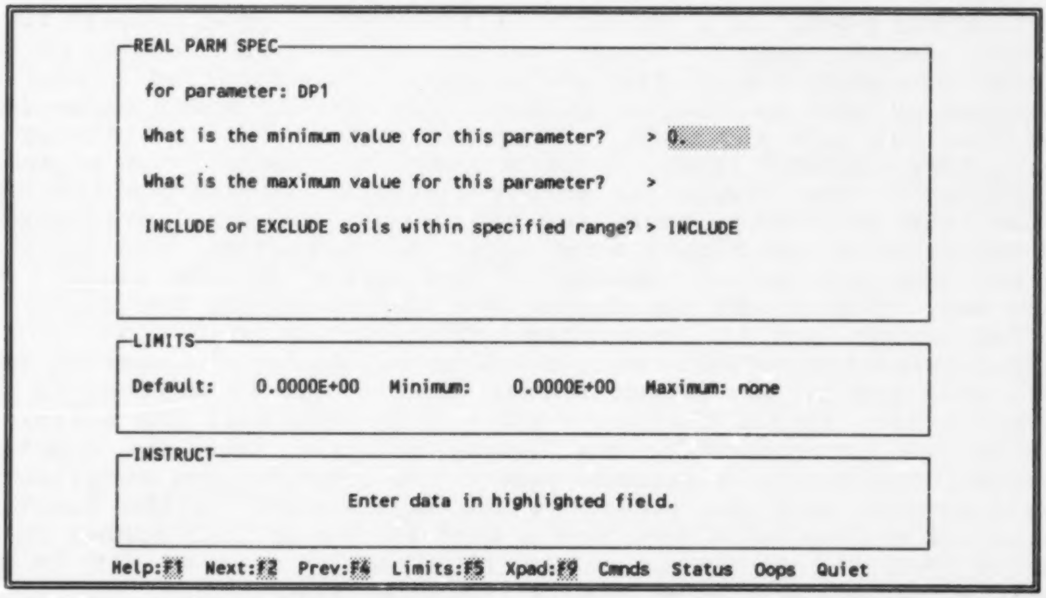
Figure 5.29. Resulting command line display.

5.14 SAVE CURRENT MENU TEXT

```
C       put name of parameter at top of menu (after blank line)
        LEN= 1
        CALL ZBLDWR (LEN,BK,I1,IM1,DONBLD)
        WRITE (TBUFX,2000) STR
        LEN= 78
        CALL ZBLDWR (LEN,TBUFF,I0,IM1,DONBLD)
C       set dont initialize menu
        CALL ZMNSST
```

Figure 5.30. Fortran code to save menu text.

Figure 5.30 shows Fortran code to control clearing of the data window. This allows the programmer to add text from the program to a menu or data editing screen. Figure 5.8 showed a screen where a minimum value, maximum value, and include/exclude option were entered. This screen is designed to define search criteria for the parameter that has just been selected from a list of parameters. This example shows how the programmer can display the parameter which was selected. Two calls to ZBLDWR are made with the write flag (fourth argument) set to a value of "-1." The result is that text is not displayed. The first call is for a blank line, with the initialization flag set to 1 (third argument), and the second call is for the parameter name. Then a call to ZMNSST sets the system to save the current text so it will be displayed on the next screen. If this code is put before the code in example 5.8, the resulting screen is Figure 5.31.



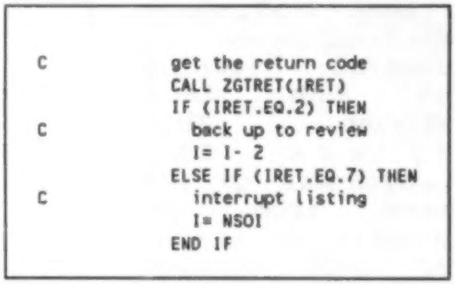
```
REAL PARM SPEC

  for parameter: DP1

  What is the minimum value for this parameter?    > 0.

  What is the maximum value for this parameter?    >

  INCLUDE or EXCLUDE soils within specified range? > INCLUDE

LIMITS

Default:    0.0000E+00  Minimum:    0.0000E+00  Maximum: none

INSTRUCT

                 Enter data in highlighted field.
```

Help:F1 Next:F2 Prev:F4 Limits:F5 Xpad:F9 Cmnds Status Oops Quiet

Figure 5.31. Resulting screen with additional text information.

5.15 DETERMINE USER SCREEN EXIT COMMAND

Figure 5.32 shows how the programmer returns the user's exit command value using a call to ZGTRET. This is useful when the programmer has made screen exits other than "NEXT" available. Screen exit codes which can be used are listed in Appendix B.7. In this example, the programmer has made the "INTRPT" and "PREV" commands available. The value of the screen exit command is checked, using conditionals, to determine whether either the "INTRPT" (value of 7) or "PREV" (value of 2) command has been selected by the user. The programmer then acts accordingly, depending on the value of the screen exit command.

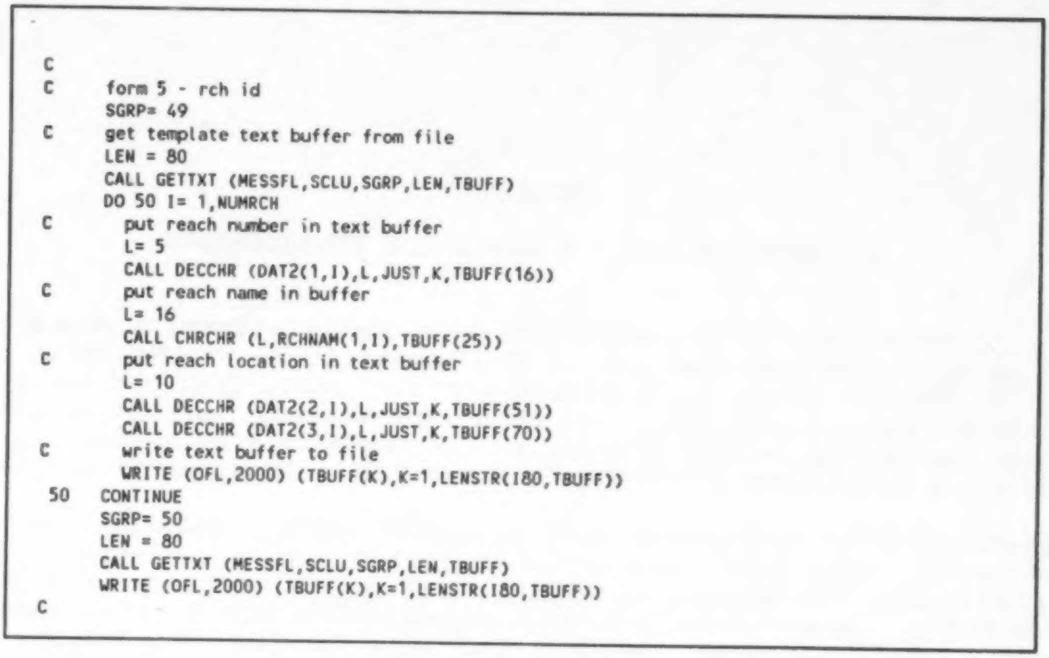


```
C          get the return code
           CALL ZGTRET(IRET)
           IF (IRET.EQ.2) THEN
C            back up to review
             I= I- 2
           ELSE IF (IRET.EQ.7) THEN
C            interrupt listing
             I= NSOI
           END IF
```

Figure 5.32. Fortran code to determine user screen exit command.

5.16 WRITE FILE CONTAINING MODEL PARAMETERS

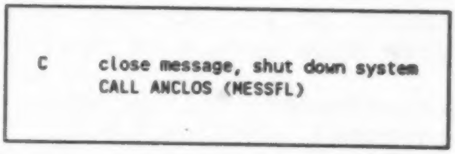
Figure 5.33 illustrates the code needed to write an output file containing a set of model parameters. The output file (OFL) has already been successfully opened. In this example, there are character strings to be output with the model parameters. The strings are retrieved from the WDM-message file using the GETTXT routine and placed in a character buffer TBUFF. The message file, screen cluster number, group number, and maximum length of the buffer are supplied by the programmer. The modified buffer is returned as well as its new length. The current model parameters are then put into the buffer using the routine INTCHR (integer to character), DECCHR (real to character), or CHRCHR (character to character). The inputs for either INTCHR or DECCHR are the data value, the available length for the value, and the justification of the value (0 for right, 1 for left) in the buffer. The modified buffer and the actual length of the value in the buffer are returned. The inputs for CHRCHR are the length of the character string arrays and the character string array to be put in the output character buffer. Once the data values for the current form have been put in the output buffer, the buffer is written to the output file. Notice that only the actual length of the buffer is output, as determined by the length returned from the function LENSTR. The buffer's maximum length and contents are supplied to the function, and the actual length is returned as the function value. The purpose of building a text string in this manner is to assure that the parameter values required by a model fit in the field width and are in an easily readable form.



```
C
C     form 5 - rch id
      SGRP= 49
C     get template text buffer from file
      LEN = 80
      CALL GETTXT (MESSFL,SCLU,SGRP,LEN,TBUFF)
      DO 50 I= 1,NUMRCH
C       put reach number in text buffer
        L= 5
        CALL DECCHR (DAT2(1,I),L,JUST,K,TBUFF(16))
C       put reach name in buffer
        L= 16
        CALL CHRCHR (L,RCHNAM(1,I),TBUFF(25))
C       put reach location in text buffer
        L= 10
        CALL DECCHR (DAT2(2,I),L,JUST,K,TBUFF(51))
        CALL DECCHR (DAT2(3,I),L,JUST,K,TBUFF(70))
C       write text buffer to file
        WRITE (OFL,2000) (TBUFF(K),K=1,LENSTR(180,TBUFF))
 50   CONTINUE
      SGRP= 50
      LEN = 80
      CALL GETTXT (MESSFL,SCLU,SGRP,LEN,TBUFF)
      WRITE (OFL,2000) (TBUFF(K),K=1,LENSTR(180,TBUFF))
C
```

Figure 5.33. Fortran code to write model parameters to a file.

5.17 SHUT DOWN SYSTEM



```
C     close message, shut down system
      CALL ANCLOS (MESSFL)
```

Figure 5.34. Fortran code to shut down ANNIE-IDE.

The code in Figure 5.34 shows a call to routine ANCLOS. This routine performs necessary functions for ending an application program's use of ANNIE-IDE. It closes the WDM-message file and writes out the scratch pad file. This call must follow all use of ANNIE-IDE and should generally be the final executable statement before the program stops.

SECTION 6

RECOMMENDATIONS FOR ANNIE-IDE ENHANCEMENTS

During the course of the ANNIE-IDE development effort, a number of ideas have surfaced that exceed the capabilities of the ANNIE and EMIFE systems which were prototypes for ANNIE-IDE. Some (for example, the scratch pad) have been implemented. Action on other ideas has been deferred in order to get feedback from system users. Ideas for additional ANNIE-IDE features fall into seven categories.

First, screen definition and movement within screens can be enhanced. Additional data could be edited within a single screen by allowing horizontal scrolling and improvements to vertical scrolling. Basic data plotting capabilities would allow better review of data values. Within 2-D data screens the ability to generate data through functional transformations of other data could assist in data editing tasks. Applications have been identified which could make use of a dynamic screen definition capability. This would allow the fields and related information for the various types of data screens to be built in a manner similar to the advanced screen building capabilities currently supported. An option to allow users to view 2-D screens as a series of 1-D screens would allow new users of an application to view more levels of assistance with each field.

Enhancements to movement between screens would benefit more advanced applications. Storage of screen movement command availability with screen definition directives would simplify use of advanced screen movement capabilities. The addition of screen directives for specifying the names of other screens that are accessible from a given screen would simplify interscreen movement.

Improvements to the scratch pad (XPAD) would allow users more control in their use of an application. Recommended improvements include allowing the user to read and write the contents of the scratch pad to a file. Also recommended is the ability to copy the contents of the data or status window to the scratch pad.

Management of user files is another area where enhancements to ANNIE-IDE could be made. Application users are sometimes asked to specify the name of a file where information was previously stored. If the file name has been forgotten, then the user must exit the application to determine the name. This inconvenience could be eliminated by making the names of files available within the application.

Another series of possible enhancements relate to "cleaner" recovery from user mistakes. Multiple levels of "OOPS" commands would allow the user to backtrack through a number of individual fields and screens. A command to interrupt an operation that was started in error without breaking out of the application entirely would allow users to continue within the application and not lose completed work.

Finally, user training could be improved by implementing an advanced tutorial facility that requires the user to enter expected keystrokes and allows for display of additional information if the user has not responded within a programmer-specified amount of time.

In addition to the specific suggestions for ANNIE-IDE enhancements found in this section, a review should be made of proposed windowing standards like X-windows with the view of incorporating ANNIE-IDE into a windowing environment.

REFERENCES

1. Badre, A.N. 1988. Usability Critique and Recommendations for the ANNIE-QUAL2E and EMIFE/123d Software. Report prepared for AQUA TERRA Consultants, Mountain View, CA.

2. Brown, L.C. and T.O. Barnwell. 1987. The Enhanced Stream Water Quality Models QUAL2E and QUAL2E-UNCAS: Documentation and User Manual. U.S. Environmental Protection Agency, Athens, GA. EPA report 600/3-87-007.

3. General Sciences Corporation. 1987. EMIFE- Environmental Model Input File Editor Fortran Utilities-Preliminary Documentation. U.S. Environmental Protection Agency, Office of Toxic Substances, Washington, DC.

4. General Sciences Corporation. 1988. PCGEMS User's Guide. Draft report prepared for U.S. Environmental Protection Agency, Office of Pesticides and Toxic Substances, Washington, DC.

5. Lumb, A.M. and J.L. Kittle, Jr. 1986. Preliminary Design of a Data Storage and Retrieval System for Hydrologic and Hydraulic Models (Watershed Data Management System, WDMS). U.S. Geological Survey, Reston, VA.

6. Lumb, A.M., K.M. Flynn and J.L. Kittle, Jr. 1988. ANNIE User's Manual. U.S. Geological Survey draft open-file report, Reston, VA.

7. Shneiderman, B. 1987. Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley, Inc., Reading, MA.

APPENDIX A. GLOSSARY OF TERMS

The following terms have specific meanings within this manual and hence require precise definition.

assistance window - four-line, middle window of standard ANNIE-IDE screen; may display help, limits, status, command definitions, scratch pad, or tutorial

command line - line at bottom of screen that displays the currently available commands

data window - top window of standard ANNIE-IDE screen displaying up to 10 or 14 lines of data (# of lines available depends on whether or not assistance window is present)

human factors - interactive program characteristics that affect user performance and satisfaction; factors include control, consistency, functionality, and response time

information type - computer-specific abbreviation for the type of ANNIE-IDE screen (valid types are: PRM1,PRM2,TEXT,MENU,FILE)

instruct window - bottom window of standard ANNIE-IDE screen containing either instructive or corrective messages; contents of window are controlled by ANNIE-IDE.

interscreen functions - program tasks related to movement between screens and other screen-independent functions

intrascreen functions - program tasks related to construction of individual screens

keyword - a character string which the ANNIE-IDE software recognizes as the beginning of a screen directive; keywords with a "$" prefix connote primary directives associated with an entire screen, and those with a "_" prefix connote subdirectives associated with a field or option

MESSAGE.SEQ file - sequential file containing screen directives for screen clusters required by all ANNIE-IDE application programs

MESSAGE.WDM file - unformatted, direct-access file used for efficient storage of user interaction text and parameter value information

MESSIE - interactive program used to import and export screen definition files into and out of the WDM-message file

routine - Fortran SUBROUTINE or FUNCTION

screen - all the information displayed on a user's monitor at one time; for ANNIE-IDE applications, the screen layout is standardized and consists of three windows and a command line

screen cluster number - identification number for a series of screen groups that are related in such a manner that they may be expected to be modified at the same time (for example, a series of screen groups related to a particular process or model subroutine;) organization as clusters saves time and space when importing and exporting and is useful for development of large programs with multiple programmers

screen definition file - formatted sequential file containing one or more screen clusters

screen directive - character strings in the screen definition file which are used to define the type and content of an ANNIE-IDE interactive screen; a screen directive consists of a keyword and associated options or information

screen group number - identification number for a series of screen directives that fully define one screen

screen type - one of five basic screen formats used by ANNIE-IDE for enabling user interaction; the five screen types are TEXT, MENU, 1-D ARRAY, 2-D ARRAY and FILE.

TERM.DAT file - formatted sequential file containing user interaction parameter values

toolkit - total ANNIE-IDE software package, including subroutines used for screen content construction, screen display control, and housekeeping tasks; also included is an interactive program to convert screen definition files to WDM-message files and vice versa

window - separate, delineated area of the screen devoted to a specific task or category of tasks; ANNIE-IDE uses a data window, an assistance window, and an instruction window

APPENDIX B. REFERENCE MANUAL

Appendix B contains detailed information about the ANNIE-IDE package. Included are:

Appendix B.1. Frequently Called Routines. Detailed information about each toolkit routine mentioned in the main body of this document. These are the routines most likely to be used by the application programmer.

Appendix B.2. Summary of all Toolkit Routines. An alphabetized list of all ANNIE-IDE toolkit routine names, the code group where they are located, and a short description of their function.

Appendix B.3. Data Structures. A description of the common blocks used internally by ANNIE-IDE which includes the definitions of the variables and where they are used.

Appendix B.4. Screen Definition Directives. A summary of all screen definition directives organized by screen type.

Appendix B.5. File Formats. Details of the XPAD.DAT, TERM.DAT and MESSAGE.WDM file formats.

Appendix B.6. TERM.DAT Parameters. A description of the TERM.DAT file parameters used by ANNIE-IDE.

Appendix B.7. ANNIE-IDE Commands. A summary of available and planned commands for ANNIE-IDE applications.

Appendix B.8. Color Codes. A table of color codes used to specify colors on PCs.

Appendix B.9. Distribution Disk Contents. A description of the ANNIE-IDE distribution disks.

APPENDIX B.1 FREQUENTLY CALLED ROUTINES

The ANNIE-IDE toolkit is designed to allow the application programmer to utilize the capabilites of over 200 utilities while dealing directly with approximatly 50 "lead" subroutines. In Section 4.1 the lead subroutines are identified and their functions are explained. This appendix provides additional programmer-oriented information including: the purpose of each module, arguments and their definitions, use of common blocks, routines called by the module, and routines that call the module.

In addition to the lead subroutines, subordinate subroutines contained in ANNIE-IDE which have been used in the programmer application examples in Section 5 are also documented in this appendix to allow better understanding of how ANNIE-IDE can implement the screens and functions which have been illustrated. Parallel documentation for all of the approximately 200 subordinate routines contained in ANNIE-IDE is available in a file called SYSDOC.OUT which is included with the ANNIE-IDE distribution disks.

This SUBROUTINE is number 2 in file ANINIT.

Close message file and scratch pad file, write out scratch pad file.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	MESSFL	I*4		I	Fortran unit number for message file

COMMON USAGE:

none

CALLS:

routine

QFCLOS ZEMSTP

CALLED BY:

unknown

This SUBROUTINE is number 1 in file ANINIT.

Initialize environment for ANNIE application.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	WDMSFL	I*4		I	Fortran unit number for WDM message file

COMMON USAGE:

block	name	status
CFBUFF	MAXREC	O

CALLS:

routine

ANPRGT COLINI GETFUN TRMINL WDBOPN WDFLCK ZEMIFE

CALLED BY:

group	routine
ANNSQ2M	ANNIE
DBAPE	DBAPE

This SUBROUTINE is number 5 in file UTCHAR.

This routine moves LEN characters from string STR1 to string STR2.

ARGUMENTS:

order	name	declaration type	size	status	explanation
1	LEN	I*4		I	length of character strings
2	STR1	C*1	(V)	I	input character string
3	STR2	C*1	(V)	O	output character string

COMMON USAGE:

none

CALLS:

none

CALLED BY:

group	routine						
ANNSQ2M	AQ2WRT						
ANNSQ2S	AQUAL2	MTIT					
ANOTHR	ANUNCA	OBSWRT	CHRCHK	CHRSRT	UNCREA	UNCWRT	VRCDHP
	GEOMAK	GSTFIN					
QTCHAR	CHKANS	PRTANS	QRESPS	QRSPIN	QTSTR		
QTFILE	QFOPFN						
QTPRIN	PMXTXM						
QTSCN1	QRESCZ	QSCSET	QUESET				
SOIMAK	SOIFSM	SOIFSP	SOIMAK	SOIPRT	GTCRNM		
UTCHAR	DATLST	DECCHX					
UTMISC	QTSTRY						
UTSORT	ASRTC						
UTWDMF	WMSGTM						
ZTSCN1	ZFLOUT						

This SUBROUTINE is number 20 in file UTCHAR.

This routine converts a real number to a character string. The number is right justified in the string if the flag JUST is zero or if the number requires an exponent, otherwise it is left justified.

ARGUMENTS:

order	name	declaration type size	status	explanation
1	REAIN	R*4	I	real value to be converted to a character string
2	LEN	I*4	I	available length for output character string
3	JUST	I*4	M	output justification 0 - right justified 1 - left justified will be forced to 0 if an exponent is required
4	JLEN	I*4	O	actual length of output character string
5	STR	C*1 (V)	O	output character string

COMMON USAGE:

none

CALLS:

routine

ABS ALOG10 CHRINS FLOAT IABS IFIX INTCHR ZIPC

CALLED BY:

group	routine			
ANNSQ2M	AQ2WRT			
ANNSQ2S	AQUAL2			
ANOTHR	ANUNCA	LCLWRT	OBSWRT	UNCWRT
GEOMAK	PRRLIS			
QTPRIN	PMXTXM			
SOIMAK	SOIFSM			
UTCHAR	PRTLIN	PRTLNO		
ZTSCN1	ZFLOUT			

This SUBROUTINE is number 1 in file QTCHAR.

Reads text from the message file into a buffer.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	message file containing string to read
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	OLEN	I*4		M	length of string read from message file
5	OBUFF	C*1	(V)	O	string read from message file

COMMON USAGE:

none

CALLS:

routine

WMSGTT

CALLED BY:

group	routine					
ANNSQ2M	AQ2WRT					
ANNSQ2S	AQUAL2					
ANOTHR	LCLWRT	OBSWRT	UNCINI	UNCWRT		
GEOMAK	ALLPRT	GEOFND	GEOMAK	GSTFIN	GSTNFI	PRRLIS
QTCHAR	QRESPS					
SOIMAK	FNDCRP	SOIFSM	SOIMAK	GTCRNM		
UTUPRM	ANPRRD					
ZUTIL	ZEMIFE	ZLIMIT				
ZUTILX	ZWRTCM					

This SUBROUTINE is number 24 in file UTCHAR.

This routine converts the integer number INTIN to a character string. The number will be right justified in the string if the flag JUST is zero, otherwise is will be left justified.

ARGUMENTS:

		declaration			
order	name	type	size	status	explanation
1	INTIN	I*4		I	integer value to be converted to a character string
2	LEN	I*4		I	available length for output character string
3	JUST	I*4		I	output justification 0 - right justified in the field 1 - left justified in the field
4	JLEN	I*4		O	actual length of output character string
5	STR	C*1	(V)	O	output character string

COMMON USAGE:

none

CALLS:

routine

ALOG10 DIGCHR INT MOD REAL ZIPC

CALLED BY:

group	routine				
ANNSQ2M	AQ2WRT				
ANOTHR	LCLWRT	OBSWRT	UNCWRT		
GEOMAK	PRRLIS				
QTPRIN	PMXTXM				
SOIMAK	SOIFSM	SOIPRT			
USCNVT	SCCUMV	SCCURM			
UTCHAR	DATCHR	DATLST	DECCHR	DECCHX	DPRCHR
ZTSCN1	ZFLOUT				

LENSTR LENSTR

This INTEGER FUNCTION is number 25 in file UTCHAR.

This routine returns the actual length of the character string, excluding trailing blanks.

ARGUMENTS:

order	name	declaration type	size	status	explanation
1	LEN	I*4		I	available length of character string
2	STR	C*1	(V)	I	character string

COMMON USAGE:

none

CALLS:

none

CALLED BY:

group	routine			
ANNSQ2M	AQ2WRT			
ANOTHR	LCLWRT	OBSWRT	CHRCHK	UNCWRT
QTCHAR	CHKANS	PRTANS	QRESPS	
QTFILE	QFOPFN			
QTNUMB	QRESPD	QRESPI	QRESPR	
QTPRIN	PMXTXM			
QTSCN1	QRESCZ			
SOIMAK	SOIFSP			
UTMISC	QTSTRY	PRTSTR		
UTUPRM	ANPRRD			
UTWDMF	WMSADI			
ZTSCN1	ZFILVL	ZFLOUT		

This SUBROUTINE is number 4 in file QTPRIN.

This routine prints text from a message file to a terminal. It prompts the user every MAXLIN lines to be sure its ok to continue. It also returns the number of lines printed.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	message file containing text to print
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	MAXLIN	I*4		I	maximum number of lines to print before more prompt
5	SCNINI	I*4		I	screen initialization flag (-1 - dont clear anything, 0 - clear in EMIFE mode only 1 - always clear)
6	LINCNT	I*4		O	count of lines printed

COMMON USAGE:

none

CALLS:

routine

PMXCNW

CALLED BY:

group	routine	
QTPRIN	PMXTXT	PRTXLC

This SUBROUTINE is number 5 in file QTPRIN.

This routine prints text from a message file to a terminal. It prompts the user every MAXLIN lines to be sure its ok to continue. It also returns the number of lines printed.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	message file containing text to print
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	MAXLIN	I*4		I	maximum number of lines to print before more prompt
5	SCNINI	I*4		I	screen initialization flag (-1 - dont clear anything, 0 - clear in EMIFE mode o 1 - always clear)
6	IWRT	I*4		I	1 - write and go, 0 - write and wait for user a -1 - dont write yet (EMIFE only)
7	LINCNT	I*4		O	count of lines printed

COMMON USAGE:

none

CALLS:

routine

ANPRGT PPRMPT SCCLR WMSGTT ZBLDWR ZTXINI

CALLED BY:

group	routine
ANNSQ2S	AQUAL2
QTPRIN	PMXCNT

This SUBROUTINE is number 13 in file QTPRIN.

This routine prints text from a message file to a terminal. It fills in values from CVAL whenever a delimeter is found. It prompts the user every MAXLIN lines to be sure its ok to continue.

ARGUMENTS:

		declaration			
order	name	type	size	status	explanation
1	UMESFL	I*4		I	message file containing text to print
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	MAXLIN	I*4		I	maximum number of lines to print before more prompt
5	SCNINI	I*4		I	screen initialization flag (-1 - dont clear anything, 0 - clear in EMIFE mode only 1 - always clear)
6	IWRT	I*4		I	1 - write and go, 0 - write and wait for user acknowledge -1 - dont write yet (EMIFE only)
7	CNUM	I*4		I	number of character strings to include in message
8	CLEN	I*4	(V)	I	array of lengths of character strings
9	CVAL	C*1	(*)	I	array containing character strings to include

COMMON USAGE:

none

CALLS:

routine
PMXTXM

CALLED BY:

group	routine
QTPRIN	PRNTXA

This SUBROUTINE is number 11 in file QTPRIN.

This routine prints text from a message file to a terminal. It fills in values from DVAL whenever delimeter is found. It prompts the user every MAXLIN lines to be sure its ok to continue.

ARGUMENTS:

order	name	declaration type size	status	explanation
1	UMESFL	I*4	I	message file containing text to print
2	SCLU	I*4	I	screen cluster number on WDM-message file
3	SGRP	I*4	I	screen group number in cluster
4	MAXLIN	I*4	I	maximum number of lines to print before more prompt
5	SCNINI	I*4	I	screen initialization flag (-1 - dont clear anything, 0 - clear in EMIFE mode only 1 - always clear)
6	IWRT	I*4	I	1 - write and go, 0 - write and wait for user acknowledge -1 - dont write yet (EMIFE only)
7	DNUM	I*4	I	number of double precision numbers to include in message
8	DVAL	R*8 (V)	I	array containing numbers to include

COMMON USAGE:

none

CALLS:

routine

PMXTXM

CALLED BY:

group	routine
QTNUMB	QRESPD
QTPRIN	PRNTXD

This SUBROUTINE is number 7 in file QTPRIN.

This routine prints text from a message file to a terminal. It fills in values from IVAL whenever delimeter is found. It prompts the user every MAXLIN lines to be sure its ok to continue.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	message file containing text to print
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	MAXLIN	I*4		I	maximum number of lines to print before more prompt
5	SCNINI	I*4		I	screen initialization flag (-1 - dont clear anything, 0 - clear in EMIFE mode o 1 - always clear)
6	IWRT	I*4		I	1 - write and go, 0 - write and wait for user a -1 - dont write yet (EMIFE only)
7	INUM	I*4		I	number of integers to include in message
8	IVAL	I*4	(V)	I	array containing integers to include

COMMON USAGE:

none

CALLS:

routine

PMXTXM

CALLED BY:

group	routine	
ANNSQ2S	AQUAL2	
QTNUMB	QRESPI	
QTPRIN	PRNTXI	
SOIMAK	SOIMAK	SOIPRT
UTDATE	CKTSTU	

This SUBROUTINE is number 14 in file QTPRIN.

This routine prints text from a message file to a terminal. It fills in values from the IVAL, RVAL, DVAL and CVAL arrays whenever a delimeter is found. It prompts the user every MAXLIN lines to be sure its ok to continue.

ARGUMENTS:

order	name	declaration type	size	status	explanation
1	UMESFL	I*4		I	Fortran unit number for WDM file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	MAXLIN	I*4		I	maximum number of lines to print before more prompt
5	SCNINI	I*4		I	screen initialization flag (-1 - dont clear anything, 0 - clear in EMIFE mode only 1 - always clear)
6	ONUM	I*4		I	number of variables to be added to the text
7	OTYP	I*4	(V)	I	array of types for variables to be added to the text
8	IWRT	I*4		I	1 - write and go, 0 - write and wait for user acknowledge -1 - dont write yet (EMIFE only)
9	IVAL	I*4	(*)	I	array containing integers to include
10	RVAL	R*4	(*)	I	array containing reals to include
11	DVAL	R*8	(*)	I	array containing double precisions to include
12	CLEN	I*4	(*)	I	array of lengths of character strings
13	CVAL	C*1	(*)	I	array containing character strings to include

COMMON USAGE:

none

CALLS:

routine

ABS ANPRGT CHRCHR CHRDEL CHRINS DECCHR DPRCHR INTCHR
LENSTR PPRMPT SCCLR STRFND WMSGTT ZBLDWR ZIPC

CALLED BY:

group	routine			
QTPRIN	PMXTXI	PMXTXR	PMXTXA	PMXTXD

This SUBROUTINE is number 9 in file QTPRIN.

This routine prints text from a message file to a terminal. It fills in values from RVAL whenever delimeter is found. It prompts the user every MAXLIN lines to be sure its ok to continue.

ARGUMENTS:

order	name	declaration type	size	status	explanation
1	UMESFL	I*4		I	message file containing text to print
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	MAXLIN	I*4		I	maximum number of lines to print before more prompt
5	SCNINI	I*4		I	screen initialization flag (-1 - dont clear anything, 0 - clear in EMIFE mode o 1 - always clear)
6	IWRT	I*4		I	1 - write and go, 0 - write and wait for user a -1 - dont write yet (EMIFE only)
7	RNUM	I*4		I	number of decimal numbers to include in message
8	RVAL	R*4	(V)	I	array containing numbers to include

COMMON USAGE:

none

CALLS:

routine
PMXTXM

CALLED BY:

group	routine
ANNSQ2S	AQUAL2
QTNUMB	QRESPR
QTPRIN	PRNTXR

This SUBROUTINE is number 2 in file QTPRIN.

This routine prints text from a message file to a terminal. It prompts the user every MAXLIN lines to be sure its ok to continue.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	message file containing text to print
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	MAXLIN	I*4		I	maximum number of lines to print before more prompt
5	SCNINI	I*4		I	screen initialization flag (-1 - dont clear anything, 0 - clear in EMIFE mode only 1 - always clear)

COMMON USAGE:

none

CALLS:

routine
PMXCNT

CALLED BY:

group	routine
QTPRIN	PRNTXT

This SUBROUTINE is number 10 in file QTPRIN.

This routine prints text from a message file to a terminal with a double precision value appended to the end of the first line.

ARGUMENTS:

order	name	declaration type size	status	explanation
1	UMESFL	I*4	I	message file containing text to print
2	SCLU	I*4	I	screen cluster number on WDM-message file
3	SGRP	I*4	I	screen group number in cluster
4	DVAL	R*8	I	array containing decimals to include

COMMON USAGE:

none

CALLS:

routine

PMXTXD

CALLED BY:

unknown

This SUBROUTINE is number 6 in file QTPRIN.

This routine prints text from a message file to a terminal with an integer value appended to the end of the first line.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	message file containing text to print
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	IVAL	I*4		I	integer value to append at end of first line

COMMON USAGE:

none

CALLS:

routine
PMXTXI

CALLED BY:

group	routine			
ANNSQ2M	AQ2REA			
ANNSQ2S	AQUAL2			
ANOTHR	ANOBDO	ANUNCA	OBSREA	UNCREA
GEOMAK	GEOMAK	GEOPRT		
QTFILE	QFOPFN			
SOIMAK	SOIMAK	SOIPRT		

This SUBROUTINE is number 8 in file QTPRIN.

This routine prints text from a message file to a terminal with an decimal value appended to the end of the first line.

ARGUMENTS:

order	name	declaration type size	status	explanation
1	UMESFL	I*4	I	message file containing text to print
2	SCLU	I*4	I	screen cluster number on WDM-message file
3	SGRP	I*4	I	screen group number in cluster
4	RVAL	R*4	I	array containing decimals to include

COMMON USAGE:

none

CALLS:

routine

PMXTXR

CALLED BY:

unknown

This SUBROUTINE is number 1 in file QTPRIN.

This routine prints text from a message file to a terminal.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	message file containing text to print
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster

COMMON USAGE:

none

CALLS:

routine

PMXTXT

CALLED BY:

group	routine				
ANNSQ2M	ANNIE	TABIND			
ANNSQ2S	AQUAL2	MPCD			
ANOTHR	ANOBDO	ANUNCA			
DBAPE	DBANAL	DBAPE	DBWRIT	SAFPAR	VADPAR
GEOMAK	GEOMAK				
QTCHAR	CHKANS	QTSTR			
QTFILE	QFOPFN				
QTNUMB	QRESPD	QRESPI	QRESPR		
QTSCN1	QRESCZ				
SOIMAK	SOIFSP	SOIMAK	SOIPRT		
UTMISC	ARRINP	QTSTRY			

This SUBROUTINE is number 1 in file QTFILE.

This routine reads file specifications from the ANNIE message file and opens the files.

ARGUMENTS:

		declaration			
order	name	type	size	status	explanation
1	MFILE	I*4		I	Fortran unit number of message file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	DAFL	I*4		O	Fortran unit number for the file opened
5	RETCOD	I*4		O	return code, 0 if file opened, else non-zero

COMMON USAGE:

none

CALLS:

routine

QFOPFN ZIPC

CALLED BY:

group	routine			
ANNSQ2S	AQUAL2			
ANOTHR	ANLOCL	ANOBDO	ANOTHR	ANUNCA
GEOMAK	GEOMAK			
SOIMAK	SOIMAK			

This SUBROUTINE is number 2 in file QTFILE.

This routine reads file specifications from the ANNIE message file and opens the files.

ARGUMENTS:

order	name	declaration type size	status	explanation
1	MFILE	I*4	I	Fortran unit number of message file
2	SCLU	I*4	I	screen cluster number on WDM-message file
3	SGRP	I*4	I	screen group number in cluster
4	FLNEXT	C*1 (64)	I	name of file to open (original QFOPEN initializes this to blank, user may supply a name from an external routine
5	PRTFLG	I*4	I	print flag, 0 - calling routine prints message, 1 - print message, dont try to fix error, 2 - print message and try to fix error
6	DAFL	I*4	O	Fortran unit number for the file opened
7	RETCOD	I*4	O	return code, 0 if file opened, else non-zero

COMMON USAGE:

none

CALLS:

routine

CARVAR CHRCHR CVARAR GETFUN IOESET LENSTR PRNTXI PRNTXT
QRESP QTSTR RECSET WMSGTF

CALLED BY:

group	routine
QTFILE	QFOPEN
UTMISC	QTSTRY

This SUBROUTINE is number 5 in file QTSCN1.

This routine prompts a user for a response that contains any combination of integer, decimal, double precision, and character strings. May be called with many rows of data and used in screen mode.

ARGUMENTS:

		declaration			
order	name	type	size	status	explanation
1	MESSFL	I*4		I	Fortran unit number of users message file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	INUM	I*4		I	number or integer responses
5	RNUM	I*4		I	number of real responses
6	DNUM	I*4		I	number of double precision responses
7	CNUM	I*4		I	number of character responses
8	NROW	I*4		I	number or rows of responses
9	SCNFG	I*4		I	indicator flag for type of input processing 0 - process one line at a time 1 - process a screen at a time
10	IVAL	I*4	(V,V)	M	array containing integer responses, calling routine may pass initial values
11	RVAL	R*4	(V,V)	M	array containing real responses, calling routine may pass initial values
12	DVAL	R*8	(V,V)	M	array containing double precision responses, calling routine may pass initial values
13	CVAL	I*4	(V,3,V)	M	array containing information about character responses (_,1) - response sequence number (_,2) - starting position of character response in TBUFF (_,3) - length of character response
14	TBUFF	C*1	(80,V)	M	array of character strings containing integer, real, and character responses in the order described by the message file

COMMON USAGE:

none

CALLS:

routine

QRESCR ZIPC

CALLED BY:

unknown

This SUBROUTINE is number 3 in file QTSCN1.

This routine prompts a user for a response that contains any combination of integer, decimal and character strings. May be called with many rows of data and used in screen mode.

ARGUMENTS:

order	name	declaration type	size	status	explanation
1	UMESFL	I*4		I	Fortran unit number of users message file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	INUM	I*4		I	number or integer responses
5	RNUM	I*4		I	number of real responses
6	CNUM	I*4		I	number of character responses
7	NROW	I*4		I	number or rows of responses
8	SCNFG	I*4		I	indicator flag for type of input processing 0 - process one line at a time 1 - process a screen at a time
9	IVAL	I*4	(V,V)	M	array containing integer responses, calling routine may pass initial values
10	RVAL	R*4	(V,V)	M	array containing real responses, calling routine may pass initial values
11	CVAL	I*4	(V,3,V)	M	array containing information about character responses (_,1) - response sequence number (_,2) - starting position of character response in TBUFF (_,3) - length of character response
12	TBUFF	C*1	(80,V)	M	array of character strings containing integer, real, a character responses in the order described by the message file

COMMON USAGE:

block	name	status
CSCREC	BLNK	A

CALLS:

routine

CSCREN QRESCX ZIPC

CALLED BY:

group	routine		
ANNSQ2S	AQUAL2	MCELFG	MPLOT
ANOTHR	ANLOCL	ANOBDO	ANUNCA
QTSCN1	QRESPX		
SOIMAK	SOIFSP		

This SUBROUTINE is number 6 in file QTSCN1.

This routine prompts a user for a response that contains any combination of integer, decimal, double precision, and character strings. Uses existing TBUFF for initial values. May be called with many rows of data and used in screen mode.

ARGUMENTS:

		declaration			
order	name	type	size	status	explanation
1	UMESFL	I*4		I	Fortran unit number of users message file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	INUM	I*4		I	number or integer responses
5	RNUM	I*4		I	number of real responses
6	DNUM	I*4		I	number of double precision responses
7	CNUM	I*4		I	number of character responses
8	NROW	I*4		I	number or rows of responses
9	SCNFG	I*4		I	indicator flag for type of input processing 0 - process one line at a time 1 - process a screen at a time
10	IVAL	I*4	(V,V)	M	array containing integer responses, calling routine may pass initial values
11	RVAL	R*4	(V,V)	M	array containing real responses, calling routine may pass initial values
12	DVAL	R*8	(V,V)	M	array containing double precision responses, calling routine may pass initial values
13	CVAL	I*4	(V,3,V)	M	array containing information about character responses (_,1) - response sequence number (_,2) - starting position of character response in TBUFF (_,3) - length of character response
14	TBUFF	C*1	(80,V)	M	array of character strings containing integer, real, and character responses in the order described by the message file

COMMON USAGE:

none

CALLS:

routine
QRESCZ

CALLED BY:

group	routine	
QTSCN1	QRESCD	QRESCX

This SUBROUTINE is number 4 in file QTSCN1.

This routine prompts a user for a response that contains any combination of integer, decimal and character strings. Uses existing TBUFF for initial values. May be called with many rows of data and used in screen mode.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	Fortran unit number of users message file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	INUM	I*4		I	number or integer responses
5	RNUM	I*4		I	number of real responses
6	CNUM	I*4		I	number of character responses
7	NROW	I*4		I	number or rows of responses
8	SCNFG	I*4		I	indicator flag for type of input processing 0 - process one line at a time 1 - process a screen at a time
9	IVAL	I*4	(V,V)	M	array containing integer responses, calling routine may pass initial values
10	RVAL	R*4	(V,V)	M	array containing real responses, calling routine may pass initial values
11	CVAL	I*4	(V,3,V)	M	array containing information about character responses (_,1) - response sequence number (_,2) - starting position of character response in TBUFF (_,3) - length of character response
12	TBUFF	C*1	(80,V)	M	array of character strings containing integer, real, a character responses in the order described by the message file

COMMON USAGE:

none

CALLS:

routine

QRESCR

CALLED BY:

group	routine
ANNSQ2S	AQUAL2
ANOTHR	ANUNCA
QTSCN1	QRESCN

This SUBROUTINE is number 7 in file QTSCN1.

This routine prompts a user for a response that contains any combination of integer, decimal, double precision, and character strings. Uses existing TBUFF for initial values. May be called with many rows of data and used in screen mode.

ARGUMENTS:

		declaration			
order	name	type	size	status	explanation
1	UMESFL	I*4		I	Fortran unit number of users message file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	INUM	I*4		I	number or integer responses
5	RNUM	I*4		I	number of real responses
6	DNUM	I*4		I	number of double precision responses
7	CNUM	I*4		I	number of character responses
8	NROW	I*4		I	number or rows of responses
9	SCNFG	I*4		I	indicator flag for type of input processing 0 - process one line at a time 1 - process a screen at a time
10	TLEN	I*4		I	length of character buffer string
11	IVAL	I*4	(V,V)	M	array containing integer responses, calling routine may pass initial values
12	RVAL	R*4	(V,V)	M	array containing real responses, calling routine may pass initial values
13	DVAL	R*8	(V,V)	M	array containing double precision responses, calling routine may pass initial values
14	CVAL	I*4	(V,3,V)	M	array containing information about character responses (_,1) - response sequence number (_,2) - starting position of character response in TBUFF (_,3) - length of character response
15	TBUFF	C*1	(V,V)	M	array of character strings containing integer, real, and character responses in the order described by the message file

COMMON USAGE:

block	name	status
CSCREC	FTYP	A
CSCREN	APOS	A
CSCREN	CCNT	A
CSCREN	CROW	M
CSCREN	FDVAL	A
CSCREN	FLEN	A
CSCREN	FLIN	A
CSCREN	NFLDS	A
CSCREN	NMHDRW	I
CSCREN	SCOL	A
ZCNTL2	RSPSTR	A
ZCNTL2	ZMNTX1	A

CALLS:

routine

CHRCHR COLOR CQRSPC CQRSPI LENSTR MOD PRNTXT WMSGTP
ZCNTL1 ZCNTL3 ZEDIT1 ZEDTOM ZFILVL ZFLOUT ZIPI ZSCDEF

CALLED BY:

group routine

ANNSQ2S MGLP MPCD MTHETA MTIT
QTSCN1 QRESCR

This **SUBROUTINE** is number 4 in file QTCHAR.

This routine prompts the user for a response with a question from the message file. Valid responses in the message file are used to check the response. The user is prompted until an acceptable answer is recieved. The responses for this routine are specific character strings.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	message file containing question
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	RESP	I*4		M	number of users response

COMMON USAGE:

none

CALLS:

routine

CFBUFF QRESPS WDNXDV WMSIDP WMSMNS ZIPC

CALLED BY:

group	routine						
ANNSQ2M	ANNIE						
ANNSQ2S	AQUAL2						
ANOTHR	ANLOCL	ANOBDO	ANOTHR	ANUNCA			
DBAPE	CORDAN	DBANAL	DBAPE	DSTAN	GSAN	MDBSAN	METAN
	SCPAN	SOILAN	DBPARM	DBWRIT	PRZCHM	PRZMET	PRZPAR
	PRZSOI	VADPAR					
GEOMAK	ALLPRT	GEOFND	GEOMAK				
QTFILE	QFOPFN						
SOIMAK	SOIFSP	SOIMAK					

This SUBROUTINE is number 6 in file QTNUMB.

This routine prompts the user for a response with a question from the message file. Valid responses in the message file are used to check the response. The user is prompted until an acceptable answer is received. The responses for this routine are double precision numbers and the valid responses are checked against a minimum and maximum that are on the message file.

ARGUMENTS:

		declaration			
order	name	type	size	status	explanation
1	UMESFL	I*4		I	message file containing question
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	DVAL	R*8		M	value of users response

COMMON USAGE:

block	name	status
CSCREC	HELP	M
CSCREN	CFLD	I
CSCREN	DDEF	A
CSCREN	DMAX	A
CSCREN	DMIN	A
CSCREN	FLEN	I
CSCREN	FLIN	I
CSCREN	SCOL	I
ZCNTL1	GPTR	A
ZCNTL1	HPTR	A
ZCNTL2	ZMNTXT	A

CALLS:

routine

ABS ANPRGT CHKDPR CHRDPR LENSTR PMXTXD PRNTXT QTSTRX
WMSGTP ZCNTL3 ZEDTOM ZLJUST

CALLED BY:

group	routine
QTNUMB	QRSPUD

This SUBROUTINE is number 2 in file QTNUMB.

This routine prompts the user for a response with a question from the message file. Valid responses in the message file are used to check the response. The user is prompted until an acceptable answer is received. The responses for this routine are integer numbers and the valid responses are checked against a minimum and maximum that are on the message file.

ARGUMENTS:

order	name	declaration type	size	status	explanation
1	UMESFL	I*4		I	Fortran unit number for WDM file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	IVAL	I*4		M	users response

COMMON USAGE:

block	name	status
CSCREC	HELP	O
CSCREN	CFLD	I
CSCREN	FLEN	I
CSCREN	FLIN	I
CSCREN	IDEF	A
CSCREN	IMAX	A
CSCREN	IMIN	A
CSCREN	SCOL	I
ZCNTL1	GPTR	A
ZCNTL1	HPTR	A
ZCNTL2	ZMNTXT	A

CALLS:

routine

ANPRGT CHKINT CRINTX LENSTR PMXTXI PRNTXT QTSTRX WMSGTP
ZCNTL3 ZEDTOM ZLJUST

CALLED BY:

group	routine		
ANNSQ2S	AQUAL2	MPLOT	
ANOTHR	ANLOCL	ANOBDO	ANUNCA
GEOMAK	GEOMAK		
QTNUMB	QRSPUI		
SOIMAK	SOIMAK		

This SUBROUTINE is number 1 in file QTSCN1.

This routine prompts a user for a response that contains any combination of integer, decimal and character strings.

ARGUMENTS:

		declaration			
order	name	type	size	status	explanation
1	UMESFL	I*4		I	Fortran unit number of users message file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	INUM	I*4		I	number or integer responses
5	RNUM	I*4		I	number of real responses
6	CNUM	I*4		I	number of character responses
7	IVAL	I*4	(V)	M	array containing integer responses, calling routine may pass initial values
8	RVAL	R*4	(V)	M	array containing real responses, calling routine may pass initial values
9	CVAL	I*4	(V,3)	M	array containing information about character responses (_,1) - response sequence number (_,2) - starting position of character response in TBUFF (_,3) - length of character response
10	TBUFF	C*1	(80)	O	character string containing integer, real, and character responses in the order described by the message file

COMMON USAGE:

none

CALLS:

routine

QRESPX ZIPC

CALLED BY:

unknown

This SUBROUTINE is number 4 in file QTNUMB.

This routine prompts the user for a response with a question from the message file. Valid responses in the message file are used to check the response. The user is prompted until an acceptable answer is received. The responses for this routine are decimal numbers and the valid responses are checked against a minimum and maximum that are on the message file.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	message file containing question
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	RVAL	R*4		M	value of users response

COMMON USAGE:

block	name	status
CSCREC	HELP	M
CSCREN	CFLD	I
CSCREN	FLEN	I
CSCREN	FLIN	I
CSCREN	RDEF	A
CSCREN	RMAX	A
CSCREN	RMIN	A
CSCREN	SCOL	I
ZCNTL1	GPTR	A
ZCNTL1	HPTR	A
ZCNTL2	ZMNTXT	A

CALLS:

routine

ABS
ANPRGT
CHKREA
CHRDEC
LENSTR
PMXTXR
PRNTXT
QTSTRX
WMSGTP
ZCNTL3
ZEDTOM
ZLJUST

CALLED BY:

group	routine
ANNSQ2M	TABIND
ANNSQ2S	AQUAL2
ANOTHR	ANUNCA
QTNUMB	QRSPUR

This SUBROUTINE is number 5 in file QTCHAR.

This routine prompts the user for a response with a question from the message file. Valid responses in the message file are used to check the response. The user is prompted until an acceptable answer is received. The responses for this routine are specific character strings.

ARGUMENTS:

order	name	declaration type size	status	explanation
1	UMESFL	I*4	I	message file containing question
2	SCLU	I*4	I	screen cluster number on WDM-message file
3	SGRP	I*4	I	screen group number in cluster
4	STRLEN	I*4	I	length of string response
5	JUST	I*4	I	justification of string (0-right, 1-left)
6	STR	C*1 (V)	M	users response string
7	STRNUM	I*4	M	number of users response

COMMON USAGE:

block	name	status
CQRSPC	OBUFF	A
CQRSPC	TANS	A
CQRSPC	UANS	A
CQRSPI	DANS	O
CQRSPI	LANS	A

CALLS:

routine

ANPRGT CHKANS CHRCHR CHRDEL CHRINS GETTXT LENSTR QTSTRX
SCCLAL STRFND WMSGTM ZCNTL1 ZCNTL2 ZCNTL3 ZIPC ZSLMNN

CALLED BY:

group	routine	
GEOMAK	GEOMAK	
QTCHAR	QRESP	
SOIMAK	SOIFSP	SOIMAK

This SUBROUTINE is number 2 in file QTSCN1.

This routine prompts a user for a response that contains any combination of integer, decimal and character strings. Uses existing TBUFF for initial values.

ARGUMENTS:

		declaration			
order	name	type	size	status	explanation
1	UMESFL	I*4		I	Fortran unit number of users message file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	INUM	I*4		I	number or integer responses
5	RNUM	I*4		I	number of real responses
6	CNUM	I*4		I	number of character responses
7	IVAL	I*4	(V)	M	array containing integer responses, calling routine may pass initial values
8	RVAL	R*4	(V)	M	array containing real responses, calling routine may pass initial values
9	CVAL	I*4	(V,3)	M	array containing information about character responses (_,1) - response sequence number (_,2) - starting position of character response in TBUFF (_,3) - length of character response
10	TBUFF	C*1	(80)	M	character string containing integer, real, and character responses in the order described by the message file, calling program may pass initial values

COMMON USAGE:

none

CALLS:

routine
QRESCN

CALLED BY:

group	routine
QTSCN1	QRESPM

This SUBROUTINE is number 5 in file QTNUMB.

This routine prompts the user for a response with a question from the message file. Valid responses are supplied by the user to check the response. The user is prompted until an acceptable answer is recieved. The responses for this routine are integer numbers and the valid responses are checked against a minimum and maximum that are supplied by the user.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	Fortran unit number for WDM file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	DMIN	R*8		I	user defined minimum value
5	DMAX	R*8		I	user defined maximum value
6	DDEF	R*8		I	user defined default value
7	DVAL	R*8		M	users response

COMMON USAGE:

none

CALLS:

routine

QRESPD QSCSET

CALLED BY:

unknown

This SUBROUTINE is number 1 in file QTNUMB.

This routine prompts the user for a response with a question from the message file. Valid responses are supplied by the user to check the response. The user is prompted until an acceptable answer is recieved. The responses for this routine are integer numbers and the valid responses are checked against a minimum and maximum that are supplied by the user.

ARGUMENTS:

order	name	declaration type size	status	explanation
1	UMESFL	I*4	I	Fortran unit number for WDM file
2	SCLU	I*4	I	screen cluster number on WDM-message file
3	SGRP	I*4	I	screen group number in cluster
4	IMIN	I*4	I	user defined minimum value
5	IMAX	I*4	I	user defined maximum value
6	IDEF	I*4	I	user defined default value
7	IVAL	I*4	M	users response

COMMON USAGE:

none

CALLS:

routine

QRESPI QSCSET

CALLED BY:

group	routine
ANNSQ2S	MPLOT
ANOTHR	ANUNCA

This SUBROUTINE is number 3 in file QTNUMB.

This routine prompts the user for a response with a question from the message file. Valid responses are supplied by the user to check the response. The user is prompted until an acceptable answer is recieved. The responses for this routine are integer numbers and the valid responses are checked against a minimum and maximum that are supplied by the user.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	Fortran unit number for WDM file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	RMIN	R*4		I	user defined minimum value
5	RMAX	R*4		I	user defined maximum value
6	RDEF	R*4		I	user defined default value
7	RVAL	R*4		M	users response

COMMON USAGE:

none

CALLS:

routine

QRESPR QSCSET

CALLED BY:

unknown

QSCSET

QSCSET

This SUBROUTINE is number 9 in file QTSCN1.

Routine to set common block values for full screen limits as definied by user (bypassing assignment from message file)

ARGUMENTS:

order	name	declaration type size	status	explanation
1	INUM	I*4	I	number of integer fields
2	RNUM	I*4	I	number of real fields
3	DNUM	I*4	I	number of double precision fields
4	CNUM	I*4	I	number of character fields
5	NMFLDS	I*4	I	number of fields for this screen
6	UIMIN	I*4 (V)	I	array of user-defined integer minimum values
7	UIMAX	I*4 (V)	I	array of user-defined integer maximum values
8	UIDEF	I*4 (V)	I	array of user-defined integer default values
9	URMIN	R*4 (V)	I	array of user-defined real minimum values
10	URMAX	R*4 (V)	I	array of user-defined real maximum values
11	URDEF	R*4 (V)	I	array of user-defined real default values
12	UDMIN	R*8 (V)	I	array of user-defined double precision minimum values
13	UDMAX	R*8 (V)	I	array of user-defined double precision maximum values
14	UDDEF	R*8 (V)	I	array of user-defined double precision default values
15	VLINFG	I*4 (V)	I	valid/invalid flag, 1- valid, 2- invalid, (0- none)
16	UCCNT	I*4 (V)	I	array of user-defined valid responses for each field
17	UCDEF	I*4 (V)	I	array of user-defined default responses for character fie
18	USTRLN	I*4 (V)	I	array of user-defined lengths of responses
19	URSPST	C*1 (960)	I	user-defined string of valid responses

COMMON USAGE:

none

CALLS:

routine						
CHRCHR	CSCREC	CSCREN	WMSPIV	ZCNTL1	ZCNTL2	ZCNTL3

CALLED BY:

group	routine		
ANNSQ2S	AQUAL2		
ANOTHR	ANOBDO		
QTNUMB	QRSPUD	QRSPUI	QRSPUR

This SUBROUTINE is number 2 in file QTCHAR.

This routine, after prompting the user, reads text from the terminal into the buffer.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	UMESFL	I*4		I	Fortran unit number for WDM file
2	SCLU	I*4		I	screen cluster number on WDM-message file
3	SGRP	I*4		I	screen group number in cluster
4	LEN	I*4		I	length of string to read
5	STR	C*1	(V)	O	string containing users response

COMMON USAGE:

block	name	status
CSCREN	FLIN	I
CSCREN	SCOL	I
ZCNTL1	GPTR	A

CALLS:

routine

ANPRGT CHRCHR CSCREC PRNTXT QTSTRX WMSGTP ZCNTL2 ZCNTL3
ZEDTOM ZLJUST

CALLED BY:

group	routine	
ANOTHR	ANOBDO	ANUNCA
QTFILE	QFOPFN	
SOIMAK	SOIFSP	

This INTEGER FUNCTION is number 29 in file UTCHAR.

This routine returns the position in the string STR of length LEN that the string FSTR of length FLEN occurs. If FSTR is not found, a zero is returned.

ARGUMENTS:

order	name	declaration type	size	status	explanation
1	LEN	I*4		I	length of character string being searched
2	STR	C*1	(V)	I	character string to be searched
3	FLEN	I*4		I	length of character string to search for
4	FSTR	C*1	(V)	I	character string to be searched for

COMMON USAGE:

none

CALLS:

none

CALLED BY:

group	routine	
ANNSQ2M	AQ2REA	
ANOTHR	LCLREA	OBSREA
QTCHAR	QRESPS	
QTPRIN	PMXTXM	
UTMISC	QTSTRY	
UTUPRM	ANPRRD	

ZBLDWR

ZBLDWR

This SUBROUTINE is number 11 in file ZUTIL.

build a screen in the menu box, options are write and wait

ARGUMENTS:

order	name	declaration type	size	status	explanation
1	OLEN	I*4		I	length of string to write
2	STRING	C*1	(V)	I	string to write
3	INIT	I*4		I	initialize screen, 0 - save old stuff, 1 - initialize, -1 - overwrite last record
4	IWRT	I*4		I	write flag, -1 - dont write, 0 - write and wait, 1 - write and dont wait
5	DONFG	I*4		O	user screen exit command

COMMON USAGE:

block	name	status
ZCNTL1	ZCLEN	A
ZCNTL1	ZCRCOL	A
ZCNTL1	ZCRLIN	A
ZCNTL1	ZCWID	A
ZCNTL1	ZHLCOL	A
ZCNTL1	ZHLLEN	A
ZCNTL1	ZHLLIN	A
ZCNTL1	ZMNLEN	A
ZCNTL2	ZMNTXT	A
ZCNTL3	ZB1N	A
ZCNTL3	ZMNCSL	M
ZCNTL3	ZMNNLI	A

CALLS:

routine

SCCUMV ZCNTER ZDRWSC ZGTKYD ZSCINI ZSTCMA ZUNCNT ZWRTMN

CALLED BY:

group	routine				
GEOMAK	ALLPRT	GEOFND	GEOMAK	GEOPRT	PRRLIS
QTPRIN	PMXCNW	PMXTXM			
SOIMAK	FNDCRP	SOIFSM	SOIFSP	SOIMAK	SOIPRT

This SUBROUTINE is number 9 in file ZUTILX.

Save xpad to a file.

ARGUMENTS:

none

COMMON USAGE:

none

CALLS:

routine

GETFUN QFCLOS ZCNTL1 ZCNTL2 ZCNTL3

CALLED BY:

group	**routine**
DBAPE	**DBAPE**

This SUBROUTINE is number 11 in file ZUTILX.

get return code from last screen

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	IRET	I*4		O	return code from last screen

COMMON USAGE:

none

CALLS:

routine		
ZCNTL1	ZCNTL2	ZCNTL3

CALLED BY:

group	routine	
ANNSQ2S	AQUAL2	
GEOMAK	PRRLIS	
SOIMAK	SOIFSP	SOIMAK

ZIPC

ZIPC

This SUBROUTINE is number 31 in file UTCHAR.

This routine fills the character array X of length LEN with the given value ZIP.

ARGUMENTS:

order	name	declaration type	size	status	explanation
1	LEN	I*4		I	length of character string
2	ZIP	C*1		I	character to fill string
3	X	C*1	(V)	O	character string to be filled

COMMON USAGE:

none

CALLS:

none

CALLED BY:

group	routine						
ANNSQ2M	AQ2INI	AQ2WRT					
ANNSQ2S	AQUAL2	MGLP	MPCD	MTHETA	MTIT		
ANOTHR	ANOBDO	ANUNCA	LCLWRT	OBSWRT	UNCINI	CHRSRT	UNCREA
	UNCWRT	VRCDHP					
QTCHAR	CHKANS	PRTANS	QRESP	QRESPS	QRSPIN		
QTFILE	QFOPEN						
QTPRIN	PMXTXM						
QTSCN1	QRESCD	QRESCN	QRESPM				
SOIMAK	SOIFSM	SOIFSP	SOIPRT	GTCRNM			
UTCHAR	DATLST	DECCHR	DPRCHR	INTCHR	PRTLNO		
UTMISC	QTSTRY						
UTUPRM	ANPRRD						
UTWDMF	WMSGTM	WMSGTP	WMSGTE				

This SUBROUTINE is number 9 in file UTNUMB.

This routine fills the double precision array X of length LEN with the given value ZIP.

ARGUMENTS:

order	name	declaration type size	status	explanation
1	LEN	I*4	I	length of array
2	ZIP	R*8	I	value to fill array
3	X	R*8 (V)	O	output array

COMMON USAGE:

none

CALLS:

none

CALLED BY:

group	routine
UTNUMB	INITMD

This SUBROUTINE is number 10 in file UTNUMB.

This routine fills the integer array X of length LEN with the given value ZIP.

ARGUMENTS:

		declaration			
order	name	type	size	status	explanation
1	LEN	I*4		I	length of array
2	ZIP	I*4		I	value to fill array
3	X	I*4	(V)	O	output array

COMMON USAGE:

none

CALLS:

none

CALLED BY:

group	routine				
ANNSQ2M	AQ2INI				
ANNSQ2S	AQUAL2				
ANOTHR	ANLOCL	ANOBDO	ANUNCA	UNCINI	UNCREA
GEOMAK	ALLPRT	GEOCLR	GEOIO		
QTSCN1	QRESCZ				
SOIMAK	SOICLR	SOIFSP	SOIMAK		
UTDATE	DATCHK				
UTNUMB	INITMD				
Z4UTIL	ZSLMNN				
ZUTIL	ZEMIFE				

This SUBROUTINE is number 11 in file UTNUMB.

This routine fills the real array X of length LEN with the given value ZIP.

ARGUMENTS:

order	name	declaration type size	status	explanation
1	LEN	I*4	I	length of array
2	ZIP	R*4	I	value to fill array
3	X	R*4 (V)	O	output array

COMMON USAGE:

none

CALLS:

none

CALLED BY:

group	routine			
ANNSQ2M	AQ2INI	AQ2REA		
ANNSQ2S	AQUAL2	MTHETA		
ANOTHR	ANLOCL	ANOBDO	ANUNCA	UNCINI
GEOMAK	ALLPRT			
SOIMAK	SOICLA	SOICLR		
UTNUMB	INITMD			

This SUBROUTINE is number 15 in file ZUTILX.

Set menu save flag.

ARGUMENTS:

none

COMMON USAGE:

none

CALLS:

routine		
ZCNTL1	ZCNTL2	ZCNTL3

CALLED BY:

group	routine	
SOIMAK	SOIFSP	SOIMAK

This SUBROUTINE is number 10 in file ZUTIL.

Write menu text to file.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	EXPFL	I*4		I	Fortran unit number of file to write to

COMMON USAGE:

none

CALLS:

routine

ZCNTL1 ZCNTL2 ZCNTL3

CALLED BY:

group	routine
SOIMAK	SOIPRT

This SUBROUTINE is number 17 in file ZUTILX.

Add to system status buffer.

ARGUMENTS:

order	name	declaration type size	status	explanation
1	STPOS	I*4	I	position in status buffer to add
2	STTXT	C*78	I	text to put in status buffer

COMMON USAGE:

block	name	status
ZCNTL1	ZSTCSL	O
ZCNTL1	ZSTLEN	M
ZCNTL1	ZSTNLN	M

CALLS:

routine

ZCNTL2 ZCNTL3 ZLNTXT ZSTCMA ZSTWRT

CALLED BY:

group	routine
GEOMAK	GEOMAK
SOIMAK	SOIMAK

This SUBROUTINE is number 18 in file ZUTILX.

Set command availability.

ARGUMENTS:

order	name	declaration type	declaration size	status	explanation
1	ICMD	I*4		I	command id number
2	IAV	I*4		I	availability code =0 - NO, 1 - YES

COMMON USAGE:

none

CALLS:

routine

ZCNTL1 ZCNTL2 ZCNTL3

CALLED BY:

group	routine				
ANNSQ2S	AQUAL2				
GEOMAK	PRRLIS				
SOIMAK	SOIFSP	SOIMAK			
UTWDMF	WMSGTM	WMSGTP	WMSGTT		
Z1UTIL	ZLOCAR				
Z4UTIL	ZSLMNN				
ZUTIL	ZCMDEX	ZEMIFE	ZBLDWR	ZQUIET	ZTXINI
ZUTILX	ZB2ON	ZDRWSC	ZSTADD		

APPENDIX B.2 SUMMARY OF ALL TOOLKIT ROUTINES

The following is a complete list of the names of all subroutines contained in the ANNIE-IDE software. A short definition of the function of each subroutine also is included. The purposes of this list are two-fold: first, to prevent the application programmer from duplicating already-used routine names when naming new subroutines developed for the specific application program; and second, to alert the programmer to the capabilities of lower-level subroutines included in the ANNIE-IDE software. The programmer may obtain further details on these subroutines in the SYSDOC.OUT file contained on the system distribution disk. For each subroutine, the appendix provides information on the subroutine type, the file in which it resides on the distribution disk, the order number indicating where the subroutine occurs within the file, and a functional description. The subroutine types are:

E - main program
S - subroutine
I - integer function
R - real function
D - double precision function

name	type	file	#	description
ANCLOS	S	ANINIT	2	Close message file and scratch pad file, write out scratch pad file.
ANINIT	S	ANINIT	1	Initialize environment for ANNIE application.
ANLOCL	S	ANOTHR	5	Routine to create local climatology data file.
ANNIE	E	ANNSQ2M	1	Process input for annie/qual2 interactively.
ANOBDO	S	ANOTHR	2	Routine to plot observed DO data.
ANOTHR	S	ANOTHR	1	For seperate files of QUAL2E, (observed DO, UNCAS and local climate data).
ANPRGT	S	UTUPRM	1	Get user parameter value, reads TERM.DAT on first entry if overlay is used.
ANPRRD	S	UTUPRM	2	Reads user specific parameters from TERM.DAT file, otherwise defaults.
ANUNCA	S	ANOTHR	8	Interaction for uncertainty analysis specifications file.
AQ2INI	S	ANNSQ2M	3	Initialize aqual2 common to default values.
AQ2REA	S	ANNSQ2M	2	Read in existing QUAL2E input sequence.
AQ2WRT	S	ANNSQ2M	4	Write out a QUAL2E input sequence.
AQUAL2	S	ANNSQ2S	1	Main subroutine for ANNIE/QUAL2.
ARRINP	S	UTMISC	6	This routine gets a set of real numbers from the user from the terminal.
ASRTC	S	UTSORT	1	Sorts character strings.
ASRTI	S	UTSORT	2	Sorts integers.
ASRTIP	S	UTSORT	3	Sorts integers in their array.
ASRTR	S	UTSORT	4	Sorts decimal numbers.
ASRTRP	S	UTSORT	5	Sorts decimal numbers in their array.
CARVAR	S	UTCHAR	1	Converts a character*1 array of size LENA to a character*LEN string.
CHDECE	R	UTCHAR	2	Converts a character string to its real equivalent.
CHINTE	I	UTCHAR	3	Converts a character string to its integer equivalent.

name	type	group	#	description
CHKANQ	S	QTCHAR	7	Checks character answers supplied by the user. Returns any conflicting answer.
CHKANS	S	QTCHAR	6	Checks character answers supplied by the user.
CHKDPR	S	UTNUMB	3	Checks the double precision DVAL against the minimum and maximum values.
CHKINT	S	UTNUMB	1	Checks the integer IVAL against the minimum and maximum values.
CHKREA	S	UTNUMB	2	Checks the real RVAL against the minimum and maximum values.
CHKSTR	I	UTCHAR	4	Searches the array of character strings STR2 for the character string STR1.
CHRCHK	S	ANOTHR	14	Searches through INVRNM array for a user input variable code.
CHRCHR	S	UTCHAR	5	Moves LEN characters from string STR1 to string STR2.
CHRDEC	R	UTCHAR	6	Converts a character string to its real equivalent.
CHRDEL	S	UTCHAR	8	Deletes the character in position POS, shifts rest of string left one position.
CHRDIG	I	UTCHAR	9	Converts a single character to an integer if the character is 0 thru 9.
CHRDPR	D	UTCHAR	7	Converts a character string to its double precision equivalent.
CHRINS	S	UTCHAR	10	Inserts the character CHAR into position COL in the string of characters STRING.
CHRINT	I	UTCHAR	11	Converts a character string to its integer equivalent.
CHRSRT	S	ANOTHR	12	Sorts variable codes from INVAR.DAT file for UNCAS section of ANNIE/QUAL2E.
CKDATE	S	UTDATE	1	Determines the calendar order of two dates.
CKNBLK	I	UTCHAR	12	Checks string for all blanks. Returns a 0 if all blanks, else it returns a 1.
CKTSTU	S	UTDATE	2	Checks time step and units for compatility.
CMPTIM	S	UTDATE	3	Compares one time code and time step to a second time code and time step.
COLGET	S	UTMISC	12	Returns the current values of the fore- and background.
COLINI	S	QTSCN1	8	Initializes the colors to be used in the program.
COLSET	S	UTMISC	11	Sets the foreground and background colors.
COPYC	S	UTCHAR	13	Copies one character array to another character array, both of length LEN.
COPYD	S	UTNUMB	4	Copies one double precision array to another the double precision array.
COPYI	S	UTNUMB	5	Copies one integer array to another integer array.
COPYR	S	UTNUMB	6	Copies one real array to another real array .
CORDAN	S	DBAPE	6	Selects coordinate analysis options.
CRINTE	I	UTCHAR	14	Converts a character string to its integer equivalent.
CRINTX	I	UTCHAR	15	Converts a character string to its integer equivalent.
CTRSTR	S	UTCHAR	16	Centers the characters within the string TITLE.
CVARAR	S	UTCHAR	17	Converts a character*LENV variable to a character*1 array of size LENA.
DATCHK	S	UTDATE	4	Checks a date (year, month, day, hour, minute, second) for valid entries.
DATCHR	S	UTCHAR	18	Puts a date string into a character string.
DATLST	S	UTCHAR	19	Puts a date/time into a character string.
DATNXT	S	UTDATE	5	Adds or subtracts a time interval from the current date and time.
DAYMON	I	UTDATE	6	Returns the number of days in a given month for a given year.
DBANAL	S	DBAPE	2	Analyzes PRZM-LMS data base.
DBAPE	E	DBAPE	1	PRZM-LMS Interactive Data Base Analyzer and Parameter Estimator (DBAPE).
DBPARM	S	DBAPE	10	Estimates parameter values using PRZM-LMS data base (DBAPE).
DBWRIT	S	DBAPE	17	Handles file writing capabilities.
DECCHR	S	UTCHAR	20	Converts a real number to a character string.
DECCHX	S	UTCHAR	22	Right justifies a real number into a character string of a given length.
DIGCHR	C	UTCHAR	23	Converts a single digit to a character.
DLIMIT	S	UTDATE	7	Finds the earliest or latest date in an array of dates.
DPRCHR	S	UTCHAR	21	Converts a double precision number to a character string.
DSTAN	S	DBAPE	8	Analyzes descriptive statistics data base.
EXRLIS	S	GEOMAK	10	Export geographic ranked list of soils.
FNDCRP	S	SOIMAK	6	Finds use of a given crop in soils database.
GBFLIS	S	SOIMAK	13	Output geographic locations and acreages.
GEOCLR	S	GEOMAK	5	Clears geographic specifications array.
GEOFND	S	GEOMAK	4	Find soils in given geographic area and summarize acres.
GEOIO	S	GEOMAK	6	Do input/output on the geographic soil data base.
GEOMAK	S	GEOMAK	1	Interact with geographic soils database.
GEOPRT	S	GEOMAK	7	Output a geographic soil summary to standard output device.
GEOTBF	S	SOIMAK	12	Look through geographic buffer to find unique soil numbers.
GETFUN	S	QTFILE	5	Gets an unused Fortran unit number or frees up a unit number for a closed file.
GETTXT	S	QTCHAR	1	Reads text from the message file into a buffer.
GSAN	S	DBAPE	4	Analyzes geographic soils data base.
GSTFIN	S	GEOMAK	9	Get the state name from the input state FIPS code.
GSTNFI	S	GEOMAK	8	Get the state FIPS code from the input state number.
GTCRNM	S	SOIMAK	10	Determines name of crop from its number off the message file.
INITM	S	UTNUMB	7	Initializes the arguments for full screen calls.

name	type	group	#	description
INITMD	S	UTNUMB	8	Initializes the arguments for full screen calls (with double precision).
INTCHR	S	UTCHAR	24	This routine converts an integer number to a character string.
IOESET	S	QTFILE	6	Set I/O error values for type of machine being used.
JDMODY	S	UTDATE	8	Converts a julian day to month and day.
LCLREA	S	ANOTHR	6	Reads in old local climatological data.
LCLWRT	S	ANOTHR	7	Writes out local climatological file.
LENSTR	I	UTCHAR	25	Returns the actual length of the character string, excluding trailing blanks.
LFTSTR	S	UTCHAR	26	Left justifies a string of characters within the buffer TITLE.
MCELFG	S	ANNSQ2S	2	Processes input for form 7 in ANNIE/QUAL2E.
MDBSAN	S	DBAPE	9	Analyzes multiple data bases.
METAN	S	DBAPE	3	Analyzes metrologic data base.
MGLP	S	ANNSQ2S	3	Processes input for form 3 in ANNIE/QUAL2E.
MPCD	S	ANNSQ2S	4	Processes input for form 2 in ANNIE/QUAL2E.
MPLOT	S	ANNSQ2S	5	Processes input for form 24 in ANNIE/QUAL2E.
MTHETA	S	ANNSQ2S	6	Processes input for form 4 in ANNIE/QUAL2E.
MTIT	S	ANNSQ2S	7	Processes input for form 1 in ANNIE/QUAL2E.
NUMPTS	S	UTDATE	9	Calculates the number of time steps between a start and end date.
OBSREA	S	ANOTHR	3	Read in observed DO data.
OBSWRT	S	ANOTHR	4	Write out observed DO data.
PCGRGT	S	USCNVT	5	Gets the graphics mode on the PC.
PCGRST	S	USCNVT	6	Sets the graphics mode on the PC.
PMXCNT	S	QTPRIN	4	PRNTXT variation, counts lines, opt to init screen.
PMXCNW	S	QTPRIN	5	PMXCNT variation, opt to: init screen, wait for user, display text.
PMXTXA	S	QTPRIN	13	Display text from message file with supplied character strings added.
PMXTXD	S	QTPRIN	11	Display text from message file with supplied double precision values added.
PMXTXI	S	QTPRIN	7	Display text from message file with supplied integer values added.
PMXTXM	S	QTPRIN	14	Display text from message file with any type(s) of supplied values added.
PMXTXR	S	QTPRIN	9	Display text from message file with supplied real values added.
PMXTXT	S	QTPRIN	2	PRNTXT variation, opt to initialize screen.
PPRMPT	S	UTMISC	5	Asks user if more text, with more lines than screen buffer, should be printed.
PRNTXA	S	QTPRIN	12	Display text from message file with a character value added to text.
PRNTXD	S	QTPRIN	10	Display text from message file with a double precisioin value added to text.
PRNTXI	S	QTPRIN	6	Display text from message file with an integer value added to text.
PRNTXR	S	QTPRIN	8	Display text from message file with an decimal value added to text.
PRNTXT	S	QTPRIN	1	Display text from message file to terminal.
PRRLIS	S	GEOMAK	3	Print ranked list of soils by acreage.
PRTANS	S	QTCHAR	8	Prints the valid character responses for a particular question.
PRTLIN	S	UTCHAR	27	This routine converts an array of real numbers into a string of characters.
PRTLNO	S	UTCHAR	28	This routine converts an array of real numbers into a string of characters.
PRTSTR	S	UTMISC	14	This routine writes a character string to the given file unit.
PRTXLC	S	QTPRIN	3	PRNTXT variation, counts the lines displayed.
PRZCHM	S	DBAPE	14	Estimates PRZM chemical data.
PRZMET	S	DBAPE	12	Estimates PRZM meteorologic data.
PRZPAR	S	DBAPE	11	Estimates data base-related PRZM parameters.
PRZSOI	S	DBAPE	13	Estimates PRZM soil parameters.
Q2BSRT	S	ANOTHR	13	Sorts the input variable names (in UNCAS) in alphabetical order.
QCHR	S	UTMISC	1	Gets a single character from a terminal.
QFCLOS	S	QTFILE	3	Closes a file and makes the Fortran unit number available.
QFOPEN	S	QTFILE	1	Reads file specifications from the message file and opens the file.
QFOPFN	S	QTFILE	2	Variation of QFOPEN, file name to be opened is specified.
QFSTAT	S	QTFILE	4	Returns a file name if the file is open or blanks if it is not open.
QRESCD	S	QTSCN1	5	Variation of QRESCN with double precision data allowed.
QRESCN	S	QTSCN1	3	Prompts user for any combination of data types (I,R,C) with multiple data rows.
QRESCR	S	QTSCN1	6	Variation of QRESCD, buffer may supply initial data values.
QRESCX	S	QTSCN1	4	Variation of QRESCN, buffer may supply initial data values.
QRESCZ	S	QTSCN1	7	Variation of QRESCR, length of buffer may be increased (>80).
QRESP	S	QTCHAR	4	Prompts user for a menu response from valid responses on the message file.
QRESPD	S	QTNUMB	6	Prompts user for a double precision response, min, max, def from message file.
QRESPI	S	QTNUMB	2	Prompts user for an integer response, min, max, def from message file.
QRESPM	S	QTSCN1	1	Prompts user for any combination of data types (I,R,C) with one data row.
QRESPR	S	QTNUMB	4	Prompts user for a real response, min, max, def from message file.
QRESPS	S	QTCHAR	5	Variation of QRESP, with the menu response returned to the calling program.

name	type	group	#	description
QRESPX	S	QTSCN1	2	Variation of QRESPM, buffer may supply initial data values.
QRSPIN	S	QTCHAR	3	Routine to set responses for QRESP (menu) type question.
QRSPUD	S	QTNUMB	5	Variation of QRESPD, user supplies minimum, maximum, and default values.
QRSPUI	S	QTNUMB	1	Variation of QRESPI, user supplies minimum, maximum, and default values.
QRSPUR	S	QTNUMB	3	Variation of QRESPR, user supplies minimum, maximum, and default values.
QSCSET	S	QTSCN1	9	Sets limits and defaults for 1 and 2-d data (bypassing message file).
QTSTR	S	QTCHAR	2	Prompts user for character value with no valid responses.
QTSTRC	S	UTMISC	2	Reads a character from the terminal.
QTSTRX	S	UTMISC	3	Reads text from terminal into buffer.
QTSTRY	S	UTMISC	4	Reads text from terminal into buffer. may read single character, or string.
QUESET	S	QTSCN1	10	Sets question for data response screens (bypassing message file).
QUPCAS	S	UTMISC	13	Converts a character string from lower case to upper case.
RECSET	S	QTFILE	7	Determine record length for an OPEN statement depending on the computer type.
SAFPAR	S	DBAPE	16	Routine to data base-related SAFTMOD parameters.
SCCLAL	S	USCNVT	1	Homes the cursor and clears the screen.
SCCLR	S	USCNVT	2	Clears the screen.
SCCUHM	S	UTMISC	8	Moves the cursor home (1,1).
SCCUMV	S	USCNVT	3	Moves cursor to the absolute row and column specified.
SCCURM	S	USCNVT	4	Moves the cursor to the relative row and column specified.
SCPAN	S	DBAPE	7	Routine to analyze soils properties and cropping practice data base.
SCPRBF	S	UTMISC	10	Builds an output string buffer and writes it out.
SCPRBN	S	USYSPR	1	Prints a string to the terminal with no cr/lf *** PRIME/PC SPECIFIC ***.
SCPRST	S	UTMISC	9	Prints a string to the terminal with no cr/lf.
SOICLA	S	SOIMAK	3	Clear out the soil area storage buffer.
SOICLR	S	SOIMAK	2	Clear out the soil storage buffer.
SOICSP	S	SOIMAK	9	Does a soil meet our parameter specifications?
SOIFSM	S	SOIMAK	8	Summarize the search being performed.
SOIFSP	S	SOIMAK	7	Specify search parameters for soil/crop search.
SOILAN	S	DBAPE	5	Select soil analysis options under the GSAN level.
SOIMAK	S	SOIMAK	1	Interacts with soils database.
SOIPRT	S	SOIMAK	5	Routine to output a soil summary to standard output device.
SOIREA	S	SOIMAK	4	Routine to input a soil properties record.
STRFND	I	UTCHAR	29	Return position in string being searched that string being searched for occurs.
STRLNX	I	UTCHAR	30	Return the length of the input string. Leading/trailing blanks not included.
TABIND	I	ANNSQ2M	5	Determine the index of a table member specified by user (ANNIE/QUAL2).
TIMADD	S	UTDATE	10	Calculates a new time from an old time and increment.
TIMBAK	S	UTDATE	11	This routine subtracts one time interval at the given units TUNIT.
TIMCHK	I	UTDATE	12	Compares two time strings. returns 1 if SDAT < EDAT.
TIMCNV	S	UTDATE	13	Converts a date/time that uses midnight convention of 00:00 to convention 24:00.
TIMCVT	S	UTDATE	14	Converts a date/time that uses midnight convention of 00:00 to convention 24:00.
TIMDIF	S	UTDATE	15	Calculates the number of timesteps between 2 dates.
TRMINL	S	UTMISC	7	Sets Fortran unit numbers for the log file and the terminal.
UNCINI	S	ANOTHR	9	Initialize common for UNCAS.
UNCREA	S	ANOTHR	10	Read old UNCAS file into UNCAS common block.
UNCWRT	S	ANOTHR	11	Write out UNCAS file.
VADPAR	S	DBAPE	15	Estimate data base-related VADOFT parameters.
VRCDHP	S	ANOTHR	15	Write out all valid variable codes in alphabetical order.
WBCWCL	I	UTWDT1	1	Calculates a WDMS block control word.
WBCWSP	S	UTWDT1	3	Breaks up a WDMS block control word.
WBCWSQ	S	UTWDT1	2	Breaks up a WDMS block control word, adjusts TSTEP and NOV if possible.
WDATCL	I	UTWDT1	4	Calculates a WDMS date (year- hour) for use at beginning of a group.
WDATSP	S	UTWDT1	5	Breaks up a WDMS date word.
WDBFIN	S	UTWDMD	12	Initializes in memory file buffer.
WDBOPN	S	UTWDMD	13	Used by batch programs to open a wdmsfl and initialize common blocks.
WDCKDS	I	UTWDMD	10	Checks existance of dataset, returns 0 - dataset doesnt exist.
WDCKDT	I	UTWDMD	9	Checks existance and type of dataset, returns 0 - dataset doesnt exist.
WDDRRC	I	UTWDMD	7	Determines WDMSFL directory record number.
WDFLCK	I	UTWDMD	2	Function checks directory of WDMSFL for major errors.
WDINIT	S	UTWDMD	11	Initializes MAXREC and buffer.
WDNXDV	S	UTWDMF	9	Move to the next data position on WDM file and return the data value.
WDNXPS	S	UTWDMF	8	Routine to get the next data position on a WDM file.
WDPRPS	S	UTWDMF	10	Routine to get the previous data position on a WDM file.

name	type	group	#	description
WDPTCL	I	UTWDT1	6	Calculates a WDMS pointer value from record number and offset within record.
WDPTSP	S	UTWDT1	7	Split up a WDMS pointer into record and offset within record.
WDRCAD	I	UTWDMD	6	Function adds number of records specified to WDMSFL.
WDRCGN	I	UTWDMD	4	Function gets a new record from free record stack.
WDRCGO	I	UTWDMD	1	Function returns the index within the in memory file buffer.
WDRCGX	I	UTWDMD	5	Gets new record from free record stack, makes new records if required.
WDRCUP	I	UTWDMD	3	Writes a record to the WDMSFL.
WDSCHK	S	UTWDMD	8	Check WDM dataset existance and type, return number of groups.
WMSADI	S	UTWDMF	3	Put message file information on WDM file.
WMSANG	S	UTWDMF	2	Add new message file group to WDM file.
WMSATS	S	UTWDT1	17	Split up message file attribute type integer parameters word.
WMSATV	I	UTWDT1	16	Calculate a word for storing attribute type message file group paramters.
WMSBCS	S	UTWDT1	13	Split up a WDM message dataset block control word.
WMSBCV	I	UTWDT1	12	Calculate a WDM message dataset block control word.
WMSBCX	S	UTWDMF	15	Return the record and offset as well as the information stored there.
WMSBTR	S	UTWDMF	13	Back up NREC text records in a text group.
WMSFBC	S	UTWDMF	16	Get first block control word in a message dataset question and its position.
WMSGTE	S	UTWDMF	12	Get one record of text off WDM file for message datasets.
WMSGTF	S	UTWDMF	7	Get message file dataset file info from WDM file.
WMSGTM	S	UTWDMF	6	Get message file dataset menu info from WDM file.
WMSGTO	S	UTWDMF	14	Starting at DREC and DPOS, get and write text until end.
WMSGTP	S	UTWDMF	4	Get message file dataset parameter info from WDM file.
WMSGTT	S	UTWDMF	5	Get message file dataset text from WDM file.
WMSIDP	S	UTWDMF	17	Given the id of a piece of group info, return the position on the WDM file.
WMSMNS	S	UTWDT1	15	Split up a message file menu type integer parameters value.
WMSMNV	I	UTWDT1	14	Calculate a message file menu type integer parameters value.
WMSP2S	S	UTWDT1	21	Split up a message file PRM2 type integer parameters value.
WMSP2V	I	UTWDT1	20	Calculate an integer word from the 5 integer parms for PRM2 class questions.
WMSPIS	S	UTWDT1	19	Split a word from WDM file and return the values stored in the word.
WMSPIV	I	UTWDT1	18	Calculates a word for storage on WDM files given two integer words.
WMSPTE	S	UTWDMF	11	Put one record of text on WDM file for message datasets.
WMSQCK	S	UTWDMF	1	Checks to see if a question already exist on WDM file.
WMSSKB	S	UTWDMF	18	Skip to end of data block on WDM file.
WTBDCL	I	UTWDT1	10	Calculates a WDMS table dimension variable.
WTBDSP	S	UTWDT1	11	Split up WDM table dimension variable.
WTBICL	I	UTWDT1	8	Calculates a WDMS table identifier from message file id, cluster and group.
WTBISP	S	UTWDT1	9	Split up WDM table identifier.
XCHECK	I	SOIMAK	11	Determines if current data base type matches type of XX soil in this record.
Z4POS	S	Z4UTIL	2	Calculate menu position for current option.
ZARCMP	S	ZOUTIL	2	Arithmetic compiler.
ZB2ON	S	ZUTILX	2	Turn on the middle box, move upper text to make it fill whats available.
ZBEEP	S	ZUTILX	1	Ring terminal bell.
ZBLDWR	S	ZUTIL	11	Build a screen in the menu box, options are write and wait.
ZCKINT	L	ZUTILX	4	Validate an integer character string.
ZCMDEX	S	ZUTIL	3	Handle command input and execution.
ZCMDIS	S	ZUTIL	5	Display current active commands and highlight default (if exists).
ZCNTER	S	ZUTILX	5	Center menu text.
ZCOLOR	S	ZUTILX	3	Control screen colors in normal and reverse video.
ZCURMV	S	ZUTIL	14	Move cursor position based on keyboard input.
ZDELET	S	ZUTILX	6	Delete a file using FORTRAN OPEN statement.
ZDRWDW	S	ZUTILX	7	Draw window on the screen, given upper left corner coordinates and window size.
ZDRWSC	S	ZUTILX	8	Draw screen for user interaction, (data window, message window, command list).
ZEDIT1	S	Z1UTIL	1	To perform menu selection for given template file.
ZEDTOM	S	ZOUTIL	1	Perform menu selections for multiple responses.
ZEMIFE	S	ZUTIL	1	Retrieve control parameters from users TERM.DAT file.
ZEMSTP	S	ZUTILX	9	Save xpad to a file.
ZFILVL	S	ZTSCN1	3	This routine fills in QRESPM arrays with the user or default value.
ZFLOUT	S	ZTSCN1	2	Fills menu text for use in EMIFE routines.
ZFMTWR	S	ZUTILX	10	Write string using a language-dependent format statement.
ZGTKEY	S	ZSYSPR	1	Wait for keyboad interrupt by performing queue I/O.
ZGTKYD	S	ZUTIL	2	Get a key stroke for the data screen.
ZGTRET	S	ZUTILX	11	Get return code from last screen.

name	type	group	#	description
ZIPC	S	UTCHAR	31	Fills character array with the input value.
ZIPD	S	UTNUMB	9	Fills double precision array with the given value.
ZIPI	S	UTNUMB	10	Fills integer array with the given value.
ZIPR	S	UTNUMB	11	Fills real array with the given value.
ZLIMIT	S	ZUTIL	12	Display limits for current field.
ZLJUST	S	ZUTILX	14	Remove leading blanks from a character string.
ZLNSTR	I	ZUTILX	12	Determine length of first word in a character string.
ZLNTXT	I	ZUTILX	13	Determine length of text in a string, (the index of last non-blank character).
ZLOCAR	S	Z1UTIL	2	Locate neighbor field according to direction of cursor movement with arrow keys.
ZLOCFD	S	Z0UTIL	3	Locate neighbor field according to direction of cursor movement.
ZMNSST	S	ZUTILX	15	Set menu save flag.
ZPRTMN	S	ZUTIL	10	Write menu text to file.
ZQUIET	S	ZUTIL	16	Turn on quiet mode.
ZSCDEF	S	ZTSCN1	1	Set default values for full screen arrays.
ZSCINI	S	ZUTILX	16	Initialize screen for a new menu.
ZSLMNN	S	Z4UTIL	1	To perform menu selection for given template file.
ZSTADD	S	ZUTILX	17	Add to system status buffer.
ZSTCMA	S	ZUTILX	18	Set command availability.
ZSTORE	S	ZUTILX	19	Store data in a buffer in specific format.
ZSTRPR	S	ZUTIL	13	Parse string of responses into string with commas seperating the responses.
ZSTWRT	S	ZUTILX	20	Display system status.
ZTRIM	S	ZUTILX	21	Remove embeded blanks between substrings in a character string.
ZTXINI	S	ZUTIL	15	Initialize parameters for start of a text screen.
ZUNCNT	S	ZUTILX	22	Uncenter menu text.
ZVERIF	S	ZUTIL	6	Validate a data value field.
ZWINDO	S	ZUTIL	17	Turn on window.
ZWRHLP	S	ZUTIL	9	Display help information.
ZWRMN1	S	Z1UTIL	3	Rewrite 2-d data screen.
ZWRSCR	S	ZUTILX	25	Write a string to specific screen position.
ZWRTB2	S	ZUTIL	7	Write a message to middle box from specified text.
ZWRTB3	S	ZUTIL	8	Write a message to bottom box from WDM file.
ZWRTCM	S	ZUTILX	23	Display a list of command descriptions.
ZWRTMN	S	ZUTILX	24	Write menu to menu window.
ZWRVDR	S	ZSYSPR	2	Write a string to a given screen position in the video reverse mode.
ZWTREA	S	ZUTILX	26	Convert floating point number to numeric string.
ZXPAD	S	ZUTIL	4	Display and modify scratch pad.

APPENDIX B.3 DATA STRUCTURES

This appendix provides detailed information on the common blocks used internally by ANNIE-IDE. Included are a general description of the common block's function, a list of parameters used to dimension arrays within the common block, and the attributes of variables and equivalences defined in the common block. Attributes defined are: type (integer, real, etc.), array dimensions, and a list of subroutines where the variable is used. For each subroutine where a common variable is used, the usage is classified as follows.

set - variable value is set
ref - variable is referenced
s/r - variable is both set and referenced
arg - variable is used as an argument to a subroutine
a/r - variable is used as an argument and referenced
asr - variable is used as an argument, referenced and set

CFBUFF

This common block contains buffers for in memory storage of records from the WDM-message file. It is found in the include file named CFBUFF.INC.

Parameters
CONREC value = 10 number of records used by in memory buffer

Variables
WIBUFF integer array (512,CONREC) in memory buffer for WDM-message file
set: WDRCGO WMSADI WMSPTE
ref: PRWMEX WDRCUP WDSCHK WDCKDT WDCKDS WMSQCK WMSANG WMSGTP WMSGTT WMSGTM WMSGTF WDNXPS WDNXDV WDPRPS WMSGTE WMSBTR
s/r: WDFLCK WDRCGN WDRCAD WDDRRC
arg: WMSBCX
a/r: WMSFBC
asr: WDBSAC
EXUSE integer reserved for future use
set: WDBFIN
RECNO integer array (CONREC) array of record numbers in WDM-message file buffer
set: WDBFIN
ref: WDRCUP WDDRRC WMSANG WDNXPS
s/r: WDRCGO
NXTPOS integer array (CONREC) next position in buffer to use
set: WDBFIN
s/r: WDRCGO
PREPOS integer array (CONREC) previous position used in buffer
set: WDBFIN
s/r: WDRCGO
FREPOS integer next free data position on current record in WDM-message file
set: WDBFIN
s/r: WDRCGO
MAXREC integer maximum number of records on the WDM-message file
set: WDBFIN
ref: WDRCGO
s/r: ANINIT WDRCAD WDINIT

COLOR

This common block is used to set and retrieve screen colors. It is found in the include file named COLOR.INC.

Variables
FRE integer foreground color for error messages
ref: ZWRTB3
FRS integer standard foreground color
ref: ZCMDIS ZWRTB3
arg: COLINI
BKE integer background color for error messages
ref: ZWRTB3
BKS integer standard background color
ref: ZCMDIS ZWRTB3
arg: COLINI
FORE integer current foreground color
set: COLSET
ref: COLGET
BACK integer current background color
set: COLSET
ref: COLGET
Equivalences
COLARR integer array (15) array equivalenced to colors for retrieval from parameter definitions
arg: COLINI

CQRSPC

This common block contains character variables used by various user response subroutines. It is found in the include file named CQRSPC.

Variables

- **QUEST** character*1 array (78) text for question in a menu type screen
  - s/r: WMSGTM
  - arg: QUESET
- **TANS** character*1 array (480) buffer containing valid responses for menu type screen
  - ref: ZSLMNN
  - s/r: WMSGTM
  - arg: QRSPIN QRESPS CHKANS
- **UANS** character*1 array (20) buffer containing users response in menu type screen
  - s/r: ZSLMNN
  - arg: QRESPS
- **OBUFF** character*1 array (80) buffer containing conflicting valid responses (from old ANNIE method)
  - a/s: QRESPS CHKANS

CQRSPI

This common block contains integer variables used by various user response subroutines. It is found in the include file named CQRSPI.INC.

Variables

- **NANS** integer number of valid responses for menu type screen
  - set: QRSPIN
  - ref: QRESPS ZSLMNN
  - s/r: WMSGTM
  - a/r: CHKANS
- **LANS** integer length of valid responses for menu type screen
  - set: QRSPIN
  - ref: ZSLMNN
  - s/r: WMSGTM
  - a/r: QRESPS CHKANS
- **DANS** integer default response for menu type screen
- **QUINIT** integer flag indicating question already set for menu type screen
  - set: QUESET
  - s/r: WMSGTM
- **RSINIT** integer flag indicating valid responses already set for menu type screen
  - set: QRSPIN
  - s/r: WMSGTM

CSCREC

This common block contains character variables used by routines which do screen i/o. It is found in the include file named CSCREC.INC.

```
Variables
  FTC      character*1   constant indicating character data field
    ref: QRESCZ ZEDTOM ZEDIT1 ZSCDEF ZFLOUT
    s/r: WMSGTP
  FTI      character*1   constant indicating integer data field
    ref: ZEDIT1 ZSCDEF ZFLOUT ZLIMIT
    s/r: WMSGTP
  FTR      character*1   constant indicating real data field
    ref: ZEDIT1 ZSCDEF ZFLOUT ZLIMIT
    s/r: WMSGTP
  FTD      character*1   constant indicating double precision data field
    ref: ZEDIT1 ZSCDEF ZFLOUT ZLIMIT
    s/r: WMSGTP
  BLNK     character*1   constant indicating blank character
    s/r: WMSGTP
    arg: ZWRMN1
    a/s: QRESCN
  HELP     character*1   constant indicating question mark
    set: WMSGTP
    s/r: QRESPI QRESPR QRESPD
  FTYP     character*1 array (30)   array indicating data field type (I,R,D,C)
    ref: QRESCZ
    s/r: WMSGTP ZSCDEF ZFLOUT ZLIMIT
    a/r: ZEDTOM ZEDIT1
  ASDSFG   character*1 array (30)   array indicating field ordering (random,ascending,descending)
    set: WMSGTP
    ref: ZEDIT1
  FDNAM    character*8 array (30)   array containing names for data fields
    s/r: WMSGTP
  FDFMT    character*8 array (30)   array containing formats for data fields
    set: WMSGTP
```

CSCREN

This common block contains numeric variables used by routines which do screen i/o. It is found in the include file named CSCREN.INC.

Variables

- NFLDS integer total number of data fields
  - ref: ZEDTOM
  - s/r: WMSGTP
  - a/r: QRESCZ
- CROW integer current data row
  - ref: WMSGTP ZWINDO
  - set: ZGTKYD
  - r/s: QRESCZ ZEDIT1 ZLOCAR
- CFLD integer current data field
  - ref: QRESPI QRESPR QRSPD ZLIMIT ZWINDO
  - s/r: WMSGTP ZLOCFD ZLOCAR
  - a/r: ZEDTOM ZEDIT1
- FPROT integer array (30) array indicating data fields' protection (NONE,CORRECT,PROTECTED)
  - ref: ZLOCAR
  - asr: WMSGTP
- SCOL integer array (30) array of data field starting columns
  - ref: QTSTR QRESPI QRESPR QRESPD ZEDTOM ZLOCFD ZEDIT1 ZLOCAR ZFLOUT
  - a/r: QRESCZ
  - asr: WMSGTP
- FLEN integer array (30) array of data field lengths
  - ref: QTSTR QRESPI QRESPR QRESPD ZLOCFD ZLOCAR ZFLOUT
  - arg: ZLIMIT
  - a/r: QRESCZ ZEDTOM ZEDIT1 ZFLOUT
  - asr: WMSGTP
- NMHDRW integer number of header rows for 2-dimensional data
  - ref: QRESCZ ZEDIT1 ZLOCAR ZWRMN1 ZFLOUT ZGTKYD
  - s/r: WMSGTP
- DTRWF integer first data row displayed in data window
  - s/r: ZEDIT1 ZLOCAR ZWRMN1
- DTRWL integer last data row displayed in data window
  - s/r: ZLOCAR ZWRMN1
- IMIN integer array (30) array of minimum values for integer data fields
  - set: QSCSET WMSGTP
  - ref: ZEDTOM ZEDIT1 ZLIMIT
  - a/r: QRESPI
- IMAX integer array (30) array of maximum values for integer data fields
  - set: QSCSET WMSGTP
  - ref: ZEDTOM ZEDIT1 ZLIMIT
  - a/r: QRESPI
- IDEF integer array (30) array of default values for integer data fields
  - set: QSCSET WMSGTP
  - ref: ZEDTOM ZEDIT1 ZSCDEF ZLIMIT
  - a/r: QRESPI
- CCNT integer array (30) array of number of valid responses for character data fields
  - set: QSCSET
  - s/r: WMSGTP
  - arg: QRESCZ ZLIMIT
- CDEF integer array (30) array of default responses for character data fields
  - set: QSCSET WMSGTP
  - ref: ZSCDEF
- RMIN real array (30) array of minimum values for real data fields
  - set: QSCSET WMSGTP
  - ref: ZEDTOM ZEDIT1 ZLIMIT
  - a/r: QRESPI
- RMAX real array (30) array of minimum values for real data fields
  - set: QSCSET WMSGTP
  - ref: ZEDTOM ZEDIT1 ZLIMIT
  - a/r: QRESPI

RDEF real array (30) array of minimum values for real data fields
set: QSCSET WMSGTP
ref: ZEDTOM ZEDIT1 ZSCDEF ZLIMIT
a/r: QRESPI
APOS integer array (30) array of pointers to data type count for each data field
set: WMSGTP
ref: ZEDTOM ZSCDEF ZFLOUT ZLIMIT
a/r: QRESCZ
SPINIT integer flag indicating min, max, and default values have already been set for data fields
set: QRESCZ QSCSET
ref: WMSGTP
FLIN integer array (30) array of line numbers in data screen text for each data field
ref: QTSTR QRESPI QRESPR QRESPD ZEDTOM ZLOCFD ZFLOUT
s/r: WMSGTP
a/r: QRESCZ
DMIN double precision array (30) array of minimum values for double precision data fields
set: QSCSET WMSGTP
ref: ZEDTOM ZEDIT1 ZLIMIT
a/r: QRESPI
DMAX double precision array (30) array of minimum values for double precision data fields
set: QSCSET WMSGTP
ref: ZEDTOM ZEDIT1 ZLIMIT
a/r: QRESPI
DDEF double precision array (30) array of minimum values for double precision data fields
set: QSCSET WMSGTP
ref: ZEDTOM ZEDIT1 ZSCDEF ZLIMIT
a/r: QRESPI
FREC integer array (30) array of record numbers to write data fields to on a file
a/s: WMSGTP
FSLOT integer array (30) array of column numbers to write data fields to on a file
a/s: WMSGTP
FDVAL integer array (30) array of pointers to start and length of valid responses for each field
set: QSCSET
ref: ZFLOUT ZLIMIT
arg: QRESCZ ZEDTOM ZEDIT1
asr: WMSGTP
FDINV integer array (30) array of pointers to start and length of invalid responses for each field
set: QSCSET WMSGTP
ref: ZLIMIT
arg: ZEDTOM ZEDIT1
WINFLG integer flag indicating user is specifying a data window in 2-dimensional data screen
set: ZWINDO
ref: ZLOCAR
s/r: ZEDIT1
WROW integer starting data row of window being defined
set: ZWINDO
ref: ZLOCAR
s/r: ZEDIT1
WFLD integer starting data field of window being defined
set: ZWINDO
ref: ZLOCAR
s/r: ZEDIT1

CUSRPM

This common block contains variables to store users parameters. It is found in the include file named CUSRPM.INC.

```
Variables
  UPRMCT   integer   count of user parameters
    ars: ANPRGT
  UPARMS   integer   array of user parameter values
    ars: ANPRGT
```

PMESFL

This parameter group contains the variable for the Fortran unit number of the WDM-message file. It is found in the include file named PMESFL.INC.

```
Parameters
  MESSFL   value = 9   Fortran unit number of WDM-message file
```

TERI

This common block contains variables which control the input file to run the program. It is found in the include file named TERI.INC.

```
Variables
  TERIFL   integer   Fortran unit number for terminal or log file
    set: TRMINL
    ref: PPRMPT ARRINP
    s/r: QTSTRY
  TEMPFL   integer   Fortran unit number for terminal file if log file in use
    set: TRMINL
    ref: QTSTRY
  LOGFL    integer   Fortran unit number for log input file
    set: TRMINL
    ref: QTSTRY ARRINP
```

ZCNTL1

This common block contains integer variables involved in screen control. It is found in the include file named ZCNTL1.INC.

```
Variables
  ZERR     integer   general error indication flag
    set: ZCMDEX ZXPAD ZWRHLP ZDRWSC ZSCINI ZSTWRT ZWRTCM
    ref: ZEDTOM ZEDIT1
    s/r: ZGTKYD ZWRTB3
  ZRET     integer   user screen exit command
    set: ZCMDEX ZSCINI
    ref: ZBLDWR
    s/r: ZEDTOM ZEDIT1 ZSLMNN
  ZMESFL   integer   Fortran unit number for screen commands and information
    set: ZEMIFE
    arg: ZLOCFD ZLOCAR ZSLMNN ZCMDEX ZLIMIT ZWRTCM
    a/r: ZWRTB3 ZDRWSC
  ZDSN     integer   cluster on WDM-message file of screen messages
    arg: ZWRTB3 ZLIMIT ZWRTCM
    a/s: ZEMIFE
  ZCMDAV   integer array (26)   array of screen command availability
    set: ZEMIFE ZSTCMA
    ref: ZLOCAR ZGTKYD ZCMDEX ZCMDIS ZWRTCM
```

```
ZFKEY    integer array (12)  array of function key command assignments
  set: ZEMIFE
  ref: ZGTKYD
ZCMDMX   integer   total number of commands currently displayed
  ref: ZCMDEX
  s/r: ZCMDIS
ZESCST   integer   escape status, indicates what affect Escape key will have on screen
  set: ZEMIFE ZXPAD ZWRTB3 ZWRHLP ZSCINI ZSTWRT ZWRTCM
  s/r: ZGTKYD
ZTUTPS   integer array (2)   position in tutorial, cluster and group on WDM-message file
  s/r: ZEMIFE
ZDTYP    integer   current type of data screen (i.e. text, menu, 1-d, 2-d)
  set: ZEDTOM ZEDIT1 ZSLMNN ZBLDWR
  ref: ZGTKYD ZCMDEX ZQUIET ZB2ON ZDRWSC
ZQUFG    integer   flag indicating 'quiet' command is available
  set: ZEMIFE ZQUIET ZB2ON ZDRWSC
  ref: ZXPAD  ZWRHLP ZLIMIT ZSTWRT ZWRTCM
ZB2LEN   integer array (4)   array of lengths of assistance window text
  set: ZSCINI
  s/r: ZXPAD
  asr: ZWRTB2
ZCLEN    integer   number of lines which data screen information has been centered
  arg: ZSLMNN ZBLDWR ZB2ON
  a/s: WMSGTM
ZCWID    integer   number of characters which data screen information has been centered
  arg: ZSLMNN ZBLDWR ZB2ON
  a/s: WMSGTM
ZMNLEN   integer array (50)   array of lengths of data window text
  set: WMSGTP ZFLOUT ZSCINI
  arg: ZSLMNN ZQUIT  ZB2ON
  a/s: WMSGTM ZEDTOM ZEDIT1
  asr: ZCMDEX ZBLDWR
ZM5LEN   integer array (3)    array of lengths of instruction window text
  set: ZSCINI
  asr: ZWRTB3
ZWN2ID   integer   indentifier for type of information in the assistance window
  set: ZEMIFE ZQUIET
  ref: ZLOCFD ZLOCAR ZSLMNN ZSCINI
  s/r: ZXPAD  ZWRHLP ZLIMIT ZDRWSC ZSTWRT ZWRTCM
ZWN3ID   integer   indentifier for type of information in the instruction window
  set: ZEDTOM ZEDIT1 ZSLMNN ZDRWSC
  s/r: ZWRTB3
ZGP3     integer   identifier for current message in instruction window
  set: ZEDTOM ZEDIT1 ZSLMNN ZDRWSC
  s/r: ZWRTB3
ZHLPLN   integer array (4)    array of lengths of help text displayed
  s/r: ZLIMIT ZWRTCM
  asr: ZWRHLP
ZMNSAV   integer   flag indicating to save current data screen text
  set: ZEDTOM ZSLMNN ZEMIFE ZMNSST
  ref: ZSCINI
ZSTLEN   integer array (10)   array of lengths of current status text
  ref: ZSTWRT
  s/r: ZSTADD
ZSTNLN   integer   current number of lines in status text
  set: ZEMIFE
  ref: ZDRWSC ZSTWRT
  s/r: ZSTADD
ZSTCSL   integer   current starting line of status text being displayed
  set: ZEMIFE
  s/r: ZSTADD ZSTWRT
ZCMDST   integer   flag indicating command line has been modified
  set: ZEMIFE ZDRWSC ZSTCMA
  s/r: ZCMDIS
```

```
ZXPLEN   integer array (10)   array of lengths of current scratch pad text
  s/r: ZEMIFE
  asr: ZXPAD
ZXPNLN   integer   current number of lines in scratch pad text
  ref: ZXPAD
  s/r: ZEMIFE
ZXPCSL   integer   current starting line of scratch pad text being displayed
  s/r: ZEMIFE ZXPAD
ZXPLIN   integer   current cursor line in scratch pad text
  s/r: ZEMIFE ZXPAD
ZXPCOL   integer   current cursor column in scratch pad text
  s/r: ZEMIFE ZXPAD
ZHLLIN   integer   current line being highlighted in data window
  set: ZSCINI
  ref: ZGTKYD ZWRHLP ZWRTMN
  s/r: WMSGTP ZEDTOM ZLOCFD ZEDIT1 ZLOCAR ZWRMN1
  arg: WMSGTM ZBLDWR ZB2ON
  asr: ZSLMNN
ZHLCOL   integer   current column being highlighted in data window
  set: ZSCINI
  ref: ZGTKYD ZWRHLP ZWRTMN
  s/r: WMSGTP ZLOCFD ZLOCAR
  arg: WMSGTM ZB2ON  ZBLDWR
  a/r: ZEDTOM ZEDIT1 ZSLMNN
ZHLLEN   integer   current length of highlight in data window
  set: WMSGTP ZLOCAR ZSCINI
  ref: ZGTKYD ZWRHLP ZWRTMN
  s/r: ZLOCFD
  arg: WMSGTM ZBLDWR ZB2ON
  a/r: ZEDTOM ZEDIT1 ZSLMNN
ZCRLIN   integer   current cursor line on screen
  set: WMSGTP ZWRMN1
  arg: ZEDTOM ZEDIT1 ZSLMNN ZXPAD ZCMDIS ZWRTB3 ZWRHLP ZSTWRT ZWRTCM ZWRTMN
  a/s: ZLOCFD ZLOCAR ZBLDWR
ZCRCOL   integer   current cursor column on screen
  set: WMSGTP
  arg: ZSLMNN ZXPAD  ZCMDIS ZWRTB3 ZWRHLP ZSTWRT ZWRTCM ZWRTMN
  a/s: ZLOCFD ZLOCAR ZBLDWR
  asr: ZEDTOM ZEDIT1
GPTR     integer   pointer to start of general help on WDM-message file
  set: WMSGTP WMSGTT WMSGTM WMSGTF ZTXINI
  ref: ZDRWSC
  arg: QRESPS ZLOCFD ZLOCAR ZSLMNN ZCMDEX
  a/r: QTSTR QRESPI QRESPR QRESPD
HPTR     integer   array of pointers to start of help for data fields on WDM-message file
  set: WMSGTP WMSGTM
  ref: ZDRWSC
  arg: QRESPS ZLOCFD ZLOCAR ZSLMNN ZCMDEX
  a/r: QRESPI QRESPR QRESPD
```

ZCNTL2

This common block contains character variables which are used to output screens. It is found in the include file named ZCNTL2.INC.

Variables

- ZMNTXT character*78 array (50) array containing the data screen text
  - set: ZSCINI
  - ref: ZPRTMN
  - s/r: WMSGTP
  - arg: QTSTR QRESCZ ZLOCFD ZLOCAR ZWRMN1 ZSLMNN ZGTKYD ZCMDEX ZWRHLP ZQUIET ZB2ON ZWRTMN
  - a/s: QRESPI QRESPR QRESPD WMSGTM ZBLDWR
  - asr: ZEDTOM ZEDIT1 ZFLOUT
- ZCMDNA character*6 array (26) array containing the screen command names
  - ref: ZCMDEX
  - arg: ZCMDIS
  - a/s: ZEMIFE
- ZHLTXT character*78 array (4) array containing the help text
  - a/s: ZWRHLP ZLIMIT ZWRTCM
- ZMSTXT character*78 array (3) array containing the instruction box text
  - a/s: ZWRTB3
- ZSTTXT character*78 array (10) array containing the status text
  - set: ZEMIFE
  - arg: ZSTADD ZSTWRT
- ZXPTXT character*78 array (10) array containing the scratch pad text
  - ref: ZEMSTP
  - a/s: ZEMIFE
  - asr: ZXPAD
- ZFNAME character*80 name of to which screen data will be output
  - set: WMSGTP
- ZFHEAD character*80 header to go at top of screen data output file
  - set: WMSGTP
- ZFTRLR character*80 trailer to go at top of screen data output file
  - set: WMSGTP
- RSPSTR character*1 array (960) buffer containing valid and invalid responses for all data fields
  - ref: ZFLOUT ZVERIF
  - s/r: WMSGTP
  - arg: QRESCZ ZLIMIT
  - a/s: QSCSET
- ZWNNAM character*8 array (13) array containing the various window names appearing on screens
  - arg: ZXPAD ZWRTB3 ZWRHLP ZLIMIT
  - a/s: ZEMIFE
  - a/r: ZSTWRT ZWRTCM
- ZSCNAM character*48 buffer for data window names specified by screen directive $WINDOW
  - set: WMSGTP WMSGTT WMSGTM
  - a/r: ZDRWSC

Equivalences

The following equivalences are needed for use in ANNIE utilities which require character*1 variables.

- ZMNTX1 character*1 array (78,50) equivalenced to ZMNTXT
  - arg: QTSTR QRESPI QRESPR QRESPD QRESCZ
  - asr: ZFLOUT
- ZCMDN1 character*1 array (6,26) equivalenced to ZCMDNA
  - ref: ZCMDEX
  - a/s: ZEMIFE
- ZHLTX1 character*1 array (78,4) equivalenced to ZHLTXT
  - set: ZWRHLP
  - arg: ZLIMIT ZWRTCM
- ZXPTX1 character*1 array (78,10) equivalenced to ZXPTXT
  - a/s: ZXPAD
- ZWNAM1 character*1 array (8,13) equivalenced to ZWNNAM
  - a/s: ZEMIFE

ZCNTL3

This common block contains integer variables invlolved in screen control, primarily in window definition. It is found in the include file named ZCNTL3.INC.

Variables

- **ZITYPE integer indicates which type of line graphics will draw the window**
  - **set: ZCOLOR**
  - **ref: ZDRWSC ZSTWRT ZWRTCM**
  - **arg: ZXPAD ZWRTB3 ZWRHLP ZLIMIT**
- **ZB1F integer screen line number of the first row of text in the data window**
  - **set: ZEMIFE**
  - **ref: ZDRWSC**
  - **arg: ZEDTOM ZEDIT1 ZSLMNN ZCMDEX ZBLDWR ZQUIET ZB2ON**
- **ZB2F integer screen line number of the first row of text in the assistance window**
  - **set: ZEMIFE**
  - **ref: ZXPAD ZWRHLP ZLIMIT ZSTWRT ZWRTCM**
  - **arg: ZWRTB2**
- **ZB3F integer screen line number of the first row of text in the instruction window**
  - **set: ZEMIFE**
  - **a/r: ZWRTB3**
- **ZB1N integer number of text rows able to be displayed in data window**
  - **set: ZEMIFE**
  - **ref: ZLOCAR ZWRMN1 ZGTKYD**
  - **s/r: ZDRWSC**
  - **arg: WMSGTM ZEDTOM ZSLMNN**
  - **a/s: ZB2ON**
  - **a/r: ZEDIT1 ZCMDEX ZBLDWR**
  - **asr: ZQUIET**
- **ZB2N integer number of text rows able to be displayed in assistance window**
  - **set: ZEMIFE ZB2ON**
  - **ref: ZXPAD ZLIMIT ZSCINI ZSTWRT ZWRTCM**
  - **s/r: ZQUIET ZDRWSC**
  - **a/r: ZWRTB2 ZWRHLP**
- **ZB3N integer number of text rows able to be displayed in instruction window**
  - **set: ZEMIFE**
  - **ref: ZSCINI**
  - **a/r: ZWRTB3**
- **ZMNNLI integer current number of text rows in data window buffer (ZMNTXT)**
  - **set: ZFLOUT ZEMIFE**
  - **ref: QRESCZ ZEDIT1 ZWRMN1 ZGTKYD ZCMDEX ZPRTMN ZDRWSC**
  - **s/r: WMSGTP ZSCINI**
  - **arg: ZB2ON**
  - **asr: WMSGTM ZBLDWR**
- **ZMNCSL integer current starting row of text displayed in data window**
  - **set: ZSCINI**
  - **ref: ZGTKYD ZQUIET ZB2ON**
  - **s/r: ZCMDEX ZBLDWR ZDRWSC**

APPENDIX B.4 SCREEN DEFINITION DIRECTIVES

This appendix provides details for all screen definition directives and subdirectives used for developing the five ANNIE-IDE screen types. Note that the keywords for directives are preceded by a "$," and those for subdirectives are preceded by a "_." Instructions for sequencing and evaluating the directives for each screen group are provided. The reader is referred to Section 5 for examples of completed screen groups.

POS indicates where to specify value for directive as follows:
1) nxt - on following records
2) aft - after directive (text)
3) afi - after directive (integer)
4) flg - flag, no value needed

ID is calculated by MESSIE and is not specified by the application programmer.

KEYWORD	POS	ID	NOTES
TEXT screen type			
$TEXT	nxt	1	text to display is data window of screen, total number of lines must be 50 or less, max length of line to display in data window is 78, use "&" to indicate spot where toolkit text screen display routine is to fill in a value
$HELP	nxt	2	general help for screen
$WINDOW	aft	19	name of screen (up to 48 characters)
MENU screen type			
$TITLE	aft	1	title of menu, maximum length 78 characters, appears on first line of data window
$DEFAULT	afi	2a	default option; if omitted, no default
$LENGTH	afi	2b	length of longest option identification string; required, must be between 1 and 63
$WIDTH	afi	2d	column width, default 78
$COLENGTH	afi	2e	length of each column, must be between 1 and 8; default 8
$OPTION	aft	3	option identification string,length must be less than or equal to $LENGTH
_DESC	aft	4	description of option, length must be less than ($WIDTH - $LENGTH - 3)
_HELP	nxt	5	help for option, use as many records as required, up to 78 characters per record
$HELP	nxt	6	general help related to all options, as many records as required, up to 78 characters per record
$WINDOW	aft	19	window name (up to 48 characters)

KEYWORD	POS	ID	NOTES
PRM1 screen type			
$WINDOW	aft	19	window name (up to 48 characters)
$SCREEN	nxt	1	up to 10 records of text which represent the screen menu. field names are preceded by '@' and may be extended by one or more '.' in order to make field wide enough for expected values, up to 30 fields may be indicated
$HELP	nxt	16	general help for screen
$FIELD	aft	2	name of field, must match name in $SCREEN (including '@', excluding '.')
_TYPE	aft	3	data type-INTEGER, REAL, DOUBLE PRECISION, or CHARACTER (type must be specified before any other subdirectives under $FIELD)
_RANGE	aft	5	minimum and maximum value allowed for numeric fields, separated by a colon, -999 if unknown
_VALID	aft	6	valid values for any type of field, separated by commas
_DEFAULT	aft	4	default value for parameter (must appear after _RANGE and _TYPE)
_INVALID	aft	7	invalid values for any type of field
_HELP	nxt	9	help information about field
*warning, the following optional keywords have not been tested within the 1-D screen routines			
_RECORD	afi	8a	record number in output data file where value will be stored
_SLOT	afi	8b	start column in output data file record where value will be stored
_PCODE	afi	10a	parameter code for field
_UCODE	afi	10b	units code for field
_FORMAT	aft	11	format for field output to file
$OUTPUT	aft	12	name of file to store parameter values
_HEADER	aft	13	character string written at beginning of output file
_TRAILER	aft	14	character string written at end of output file
PRM2 screen type			
$WINDOW	aft	19	window name (up to 48 characters)
$HEADER	nxt	16	header for screen (may be multiple lines)
$HELP	nxt	15	general help for screen
$FIELD	aft	2	name of input array field (no '@' required as in 1-D)
*next 5 directives required before other directives			
_TYPE	aft	3a	data type- INTEGER, REAL, DOUBLE PRECISION, or CHARACTER
_WIDTH	aft	3b	field width, in characters
_ORDER	aft	3c	sort requirement - RANDOM (default), ASCENDING, DESCENDING
_PROTECT	aft	3d	field protection - NONE (default), CORRECT (must be in valid range), PROTECTED (cannot change value)
_COLUMN	afi	3e	starting column of field
_HELP	nxt	9	help information for field
_RANGE	aft	5	valid range for numeric field
_VALID	aft	6	valid values for any type field
_DEFAULT	aft	4	default value for field
_INVALID	aft	7	invalid parameter values for any type
*warning, the following optional keywords have not been tested within the 1-D screen routines			
_PACK	afi	8a	number of values to pack on a data record
_FORMAT	afi	8b	format of values on data record
_PCODE	afi	10a	parameter code for field
_UCODE	afi	10b	units code for field
_FORMAT	aft	11	format for field output to file
$OUTPUT	aft	12	name of file to store parameter values on
_HEADER	aft	13	character string written at beginning of output file
_TRAILER	aft	14	character string written at end of output file

KEYWORD	POS	ID	NOTES
FILE type screen			
$WINDOW	aft	19	window name (up to 48 characters)
$SCREEN	aft	1	screen menu text when asking for name of file, up to 10 records, field name preceded by '@' and may be extended by '.', this keyword used only if file name is to be provided by user
$NAME	aft	2	name of file ($SCREEN keyword not allowed) provided by application or field name in screen text directive which will be provided by user
$STATUS	aft	3	file status, OLD (default), NEW, UNKNOWN, or SCRATCH
$ACCESS	aft	4	file access method, SEQUENTIAL (default) or DIRECT
$FORM	aft	5	file format, FORMATTED (default) or UNFORMATTED
$RECL	aft	7	length of record in direct access file (bytes if formatted, half words if unformatted)
$HELP	nxt	6	help related to file

APPENDIX B.5 FILE FORMATS

This appendix provides format details for files that must be utilized by an application programmer who intends to use the ANNIE-IDE software. The files are introduced and their purpose and contents are discussed in Section 4.3. Additional details for the TERM.DAT file are provided in Appendix B.6.

TERM.DAT

The TERM.DAT file is a formatted, sequential file that may be constructed and manipulated with a text editor. One parameter name and its value are to be specified per line. The parameter name should be followed by one blank and then the desired value. Blank lines at the top or bottom of the file may cause problems on certain systems. The number of lines in the file is limited only by the number of available parameters for which the user wishes to specify values.

XPAD.DAT

The scratch pad file is a formatted, sequential file that may be constructed and manipulated with a text editor. The maximum number of lines is ten, and the length of these lines must not be greater than 78.

Screen Definition

The screen definition file is a formatted, sequential file that may be constructed and manipulated with a text editor. The length of lines may not be greater than 120. However, the length is most often limited by the maximum length of the variable being defined (e.g. maximum length of 78 for any data to be displayed in any of the windows). There is no column dependency for the values being defined, although spacing between keywords and their values is suggested for readablility. Careful attention should be paid to correct spelling of keywords for directives and sub-directives as the whole word is searched for.

WDM-message

The WDM-message file is an unformatted direct access file with record lengths of 2048 bytes. This file must not be manipulated in any way with a text editor or Fortran READ or WRITEs. It must only be created or modified using MESSIE.

APPENDIX B.6 TERM.DAT PARAMETERS

This appendix describes the parameters of the user system specifications file (TERM.DAT), which is introduced in Section 4.3. Information provided includes: parameter code numbers (used by programmer), code names (required by program user), default values, allowable values, and definitions. Additional TERM.DAT parameters are found in the ANNIE users manual (Lumb et al., 1988.)

PARAMETER CODE NUMB AND NAME	DEFAULT VALUE	ALLOWABLE VALUES	DEFINITION
1 CMPTYP	PC	PC PRIME VAX	type of computer, use PC for IBM clones
2 TRMTYP	PC	PC VT100	type of terminal
3 TRMINP	1	0 to 10	Fortran unit number for reading from the terminal
4 TRMOUT	1	0 to 10	Fortran unit number for writing to the terminal: 1 for PRIME, 6 for DEC VAX, 6 for PC, 0 for Data General.
5 TRMPAU	124	33 to 127	pause character, sequence number in character set
6 TRMFIL	64	33 to 127	fill character, sequence number in character set
11 SCRWID	80	40 to 256	number of columns on the terminal screen
12 SCRLEN	24	10 to 100	number of lines shown on the terminal screen
13 FILMAX	15	10 to 99	maximum number of file units open at any one time
14 FILUNI	30	7 to 99	first unit number to assign to files needed by ANNIE-IDE
15 RECTYP	WORD	BYTE WORD	units for specifying record lengths on OPEN statements for unformatted direct access files.
color definitions (see Appendix B.7 for values for PC)			
22 CLRFRM	15	0 to 15	foreground color for modified data (currently as CLRFRS)
26 CLRFRE	4	0 to 15	foreground color for error messages
28 CLRFRD	15	0 to 15	foreground color for data window (currently same as CLRFRS)
30 CLRFRS	7	0 to 15	standard foreground color
32 CLRBKE	2	0 to 15	background color for error messages
33 CLRBKS	0	0 to 15	standard background color

PARAMETER CODE NUMB AND NAME	DEFAULT VALUE	ALLOWABLE VALUES	DEFINITION
35 BUGFLG	NONE	NONE,SOME LOTS	amount of debug printout
45 USRLEV	0	0 or 2	User experience level 0-lots, 2=none

function key definitions

91 PFKEY1	H		Allowable values are the first letters of system commands supported. The PFKEY3 is defaulted to no value as it is needed to emulate ESC when using a mouse.
92 PFKEY2	N		
93 PFKEY3	-		
94 PFKEY4	P		
95 PFKEY5	L		
96 PFKEY6	-		
97 PFKEY7	S		
98 PFKEY8	Q		
99 PFKEY9	X		

APPENDIX B.7 ANNIE-IDE COMMANDS

This appendix describes the commands which are currently available in, or are planned extensions of, the ANNIE-IDE development system. First, the eleven basic commands are described, followed by three planned extensions. See Section 3.7 for additional discussion of these fourteen commands. Finally, the seven optional commands which can be used to implement movement between screens are described. See Section 3.10 for additional discussion of these commands.

BASIC APPLICATION COMMANDS

COMMAND NAME	COMMAND ORDER #	COMMAND DEFINITION
CMND	2	DISPLAY DEFINITIONS OF VALID COMMANDS IN ASSISTANCE INFORMATION WINDOW
DNPG	14	DISPLAY NEXT PAGE IN DATA WINDOW
HELP	1	DISPLAY HELP INFORMATION IN ASSISTANCE INFORMATION WINDOW
LIMITS	6	DISPLAY LIMITS OF CURRENT FIELD IN ASSISTANCE INFORMATION WINDOW
NEXT	3	GO TO NEXT SCREEN (SETS SCREEN EXIT STATUS CODE TO 1)
OOPS	8	RESET VALUES IN DATA WINDOW TO VALUES WHEN SCREEN FIRST DISPLAYED
QUIET	20	TURN OFF ASSISTANCE INFORMATION WINDOW TO ALLOW MORE ROOM FOR DATA
STATUS	7	DISPLAY SYSTEM STATUS IN ASSISTANCE INFORMATION WINDOW
UPPG	15	DISPLAY PREVIOUS PAGE IN DATA WINDOW
WINDOW	9	DEFINE CORNER OF DATA OPERATION WINDOW
XPAD	21	DISPLAY USERS SCRATCH PAD, ALLOW CHANGES
planned command extensions		
MOVE	10	COPY ROWS/COLUMNS OF TWO-DIMENSIONAL DATA
TUTOR	5	PROVIDE TUTORIAL
VIEW	11	DISPLAY GRAPHIC DATA

OPTIONAL COMMANDS FOR MOVEMENT BETWEEN SCREENS

COMMAND NAME	COMMAND ID #	SCREEN EXIT STATUS CODE	COMMAND DEFINITION
ABORT	18	6	STOP EVERYTHING QUICKLY
BEGIN	12	3	GO TO FIRST SCREEN
EXIT	17	5	CLEAN UP AND RETURN TO OPERATING SYSTEM
GOTO	13	4	GO TO SCREEN TO BE SPECIFIED
INTRPT	16	7	STOP CURRENT ACTIVITY
NEXT	3	1	GO TO NEXT SCREEN (ALWAYS AVAILABLE)
PREV	4	2	GO TO PREVIOUS SCREEN

APPENDIX B.8 COLOR CODES FOR PC COLOR IMPLEMENTATION

The following color code numbers are acceptable values for the color specifications parameters in the TERM.DAT file (i.e., parameters CLRFRM, CLRFRE, CLRFRD, CLRFRS, CLRBKB, CLRBKS) in Appendix B.6. Note that color specification is only meaningful for program applications using PCs with color monitors.

NUMBER	DESCRIPTION
0	BLACK
1	BLUE
2	GREEN
3	CYAN
4	RED
5	MAGENTA
6	BROWN
7	LIGHT GRAY
8	DARK GRAY
9	BRIGHT BLUE
10	BRIGHT GREEN
11	BRIGHT CYAN
12	BRIGHT RED
13	BRIGHT MAGENTA
14	YELLOW
15	WHITE

APPENDIX B.9 DISTRIBUTION DISK CONTENTS

The distribution diskettes for ANNIE-IDE are organized into four groups. These are: utilities, MESSIE, DBAPE and QUAL2E, and documentation. Each diskette, and any directory on it, contains a READ.ME file describing each file in more detail.

The utilities diskette contains all of the source code for ANNIE-IDE. It also contains an object library of routines (ANNIEIDE.LIB) which was compiled using Ryan-McFarland Fortran. This library allows PC application programmers to skip the step of compiling the utility routines. The utility diskette also contains a number of directories which contain system-specific code. When setting up ANNIE-IDE on a machine, follow the appropriate directory paths to use the correct system-specific code groups.

The MESSIE diskette contains the source code for MESSIE and the executable version of the program. This diskette also contains the general screen definition file (MESSAGE.SEQ) required for all ANNIE-IDE applications and a template screen definition file (SDEF.TEM).

The DBAPE and QUAL2E diskettes contain the source code for the two applications as well as the executables for a PC. Each application has its own screen definition file. The DBAPE diskettes also contain databases which are used by the program. Three of the databases (SOIML.DAT,SOIXX.DAT,GEOML.DAT) are provided in both sequential and direct access formats; the direct access files can be used directly by the program. The remaining databases (*.INF) are all in sequential format, and they must be converted to direct access by using the CREATE program (also supplied on these diskettes) before they can be used by the DBAPE program. The contents of these databases are described in more detail in the READ.ME files on the DBAPE diskettes. If DBAPE is to be run on a machine other than a PC, the sequential versions of all the databases must be transferred to the machine and then made direct access by using the CREATE program.

The documentation diskette contains a copy of the ANNIE-IDE manual. It also contains a complete listing of SYSDOC.OUT as referenced at the beginning of Appendix B.1.

INDEX

$ACCESS 44,67,157
$COLENGTH 41,155
$DEFAULT 41,66,155
$FIELD 42,43,67,69,71,73,156
$FORM 44,67,157
$HEADER 43,47,71,73,156
$HELP 40-44,47,62,63,66,67,73,155-157
$LENGTH 41,66,155
$NAME 44,67,157
$OPTION 41,47,66,155
$OUTPUT 42,43,156
$RECL 44,157
$SCREEN 42,44,47,67,69,156,157
$STATUS 44,67,157
$TEXT 40,47,62-64,155
$TITLE 41,66,155
$WIDTH 41,155
$WINDOW 40-44,62-64,66,69,73,153,155-157
ABORT 35,46,161
ANCLOS 79,87,138
ANINIT 60,87,88,138,145
assistance window 22,24-28,30,32,47,59,83,84,151,154
BEGIN 2,35,46,60,161
CHRCHR 72,78,79,89,99,107,113,119,124,125,139
cluster 37-40,46,47,50,62,64,65,67,68,78,84,91,94-112, 114-123,125,142,150,151
CMND 161
command line 21,22,24,25,28-30,33,35,76,83,84,151
data operation window 29,161
data window 21,22,24-27,29-32,40,41,47,52,62,64,77, 83,84,142,148,149,151,152,153-155,159,161
dataset 37,141,142
DECCHR 72,78,79,90,92,99,130,139
DNPG 27,29,31,161
EXIT 29,30,35,53,59,78,80,127,150,161
FILE 2-5,7-9,11,12,13,15,21,33-40,42-53,55,57,58,60,61, 62,64-69,74,78-80,82-112,114,115-143,145-154, 156-159,162,163
GETTXT 78,79,91,119,139
GOTO 35,161
group 7,15,32-34,37-39,49-51,56,61-73,76,78,84,85,88-137,

141,142,150,151,155
HELP 1,9-12,19,20,25,26,28-30,32,34,40,41,42-44,47,62-64, 66-74,77,83,115,116,118,143,147,151-153,155-157, 161
information type 83
initialize 8,48,49,53,60-62,64,72,77,88,127,138,140,141,143
instruct window 28,83
INTCHR 78,90,92,99,130,140
interscreen functions 21,83
intrascreen functions 13,21,83
INTRPT 35,76,78,161
LENSTR 78,79,93,99,107,113,115,116,118,119,140
LIMITS 9-11,25,26,28-30,58,68-70,72,74,77,83,124,141,143, 161
MENU 2,6,8,10-12,18,19,20,24-26,28,30-34,38,41,42,44,47,49, 50,53,54,57,65,66,77,83,84,127,134,135,140-143, 146,151,155,156,157
MESSAGE.SEQ 38,40,45,46,83,163
MESSAGE.WDM 36-39,45-48,52,58,83,85
MESSIE 7,33,37,38,45-47,57,84,155,158,163
MOVE 8,9,14,29,31,32,35,141,142,161
NEXT 1,8-11,20,28-30,32,35,43,45,50,58,62-66,68,70-72, 74-78,141,145,156,161
OOPS 10,11,29,31,68,70,72,74,77,81,161
PMXCNT 52,53,94,95,101,140
PMXCNW 52,53,94,95,127,140
PMXTXA 52,53,96,99,140
PMXTXD 52,53,97,99,102,115,140
PMXTXI 52,53,64,98,99,103,116,140
PMXTXM 52,53,89,90,92,93,96-100,126,127,130,140
PMXTXR 52,53,99,100,104,118,140
PMXTXT 52,53,94,101,105,140
PREV 10,35,70,76-78,161
PRM1 38,42,47,67,69,83,156
PRM2 38,43,47,69,71,73,83,142,156
PRNTXD 52,53,97,102,140
PRNTXI 52,53,98,103,107,140
PRNTXR 52,53,100,104,140
PRNTXT 52,53,62,101,105,107,113,115,116,118,125,140
QFOPEN 57,67,106,107,130,140
QFOPFN 57,89,93,103,105-107,114,125,140
QRESCD 54-56,108,110,130,140
QRESCN 54-56,68,70,72,109,111,120,130,140,147
QRESCR 54-56,108,110,111,113,140
QRESCX 54-56,72,73,109-111,140
QRESCZ 54-56,89,93,105,110,112,132,140,147-149,153,154
QRESP 54,65,107,114,119,130,140,141
QRESPD 54,55,93,97,105,115,121,140,141,147,148,149,152,153
QRESPI 54,55,93,98,105,116,122,140,141,147,148,149,152,153
QRESPM 54,55,117,120,130,140-142
QRESPR 54,55,93,100,105,118,123,140,141,147,148,149,152,153

QRESPS 54,89,91,93,114,119,126,130,140,146,152
QRESPX 54,55,109,117,120,141
QRSPUD 54,55,115,121,124,141
QRSPUI 54,55,116,122,124,141
QRSPUR 54,55,118,123,124,141
QSCSET 74,89,121-124,141,148,149,153
QUIET 9-11,29,30,63,68,70,76,77,143,151,161
screen definition file 3,33,37-39,45,46,57,66,69,74,84,
158,163
screen directive 34,60,67,83,84,153
STATUS 7,8,10,25-27,29,30,34-36,44,59,67,70,75-77,80,83,
87-112,114,115,116-127,129-133,135-137,143,151,
153,157,161
STRFND 61,62,99,119,126,141
structure chart 51,52,54-57
system status 8,26,27,29,36,59,75,136,143,161
TERM.DAT 48,58,60,84,85,138,142,158,159,162
TEXT 2,7-9,12,13,21,26,27,30,32-34,36-38,40-45,47,52,53,
54-56,60,62-65,67,75,77-79,83,84,91,94-105,125,
135,136,139-143,146,149,151-158
toolkit 2-4,6-9,12-14,16,20,21,33,36,37,40,48-51,54,59,60,
84-86,138,155
TUTOR 25-27,29,30,161
UPPG 27,29,31,161
VIEW 8,29,31,52,80-82,161
window 9,11,21,22,24-32,40-44,47,52,59,62-64,66,69,72-74,
77,80,83,84,142,143,148,149,151-157,159,161
XPAD 9-11,25-27,29,30,32,48,50,63,65,66,68,70,72,74,76,77,
80,85,128,142,158,161
ZBLDWR 52,53,77,95,99,127,137,142,150-154
ZEMSTP 87,128,142,153
ZGTRET 59,78,129,142
ZIPC 61,90,92,99,106,108,109,114,117,119,130,143
ZIPD 61,131,143
ZIPI 61,74,113,132,143
ZIPR 61,74,133,143
ZMNSST 77,134,143,151
ZPRTMN 135,143,153,154
ZSTADD 59,75,136,137,143,151,153
ZSTCMA 59,76,127,136,137,143,150,151

✩U.S. GOVERNMENT PRINTING OFFICE: 1989 648-163/87104